Republic of Iraq
Ministry of Higher Education
and Scientific Research
University of Babylon
College of Education for
Pure Sciences
Department of Mathematics

Function Approximation by MLP Algorithm

A Project

Submitted to the Council of College of Education for Pure Sciences in University of Babylon in Partial Fulfillment of the Requirement for the Degree of Higher Diploma Education/ Mathematics.

BY:

Ahlam Mohammed Abed Hamza

Supervised by:

Dr. Hawraa Abbas Almurieb

2021 A.M 1443 A.H

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

وَيَسۡـَٔلُونَكَ عَنِ ٱلرُّوحِۖ قُلِ ٱلرُّوحُ مِنۡ أَمۡرِ رَبِّى وَمَآ أُوتِيتُم مِّنَ ٱلۡعِلۡمِ إِلَّا قَلِيلًا

صدق الله العلّي العظيم

الآية (85)من سورة الاسراء

Acknowledgments

Alhamdulillah, for everything, for giving me the right people who support me to the best. Thank you for the special one who is very kind with me , my dear doctor, the amazing supervisor:

(Hawraa Abass ALmurieb)

Dedication

(I dedicate this humble work to Imami Sahib AL-Zaman AL-Huja AL-Muntadar (عج)

Abstract

The approximation by neural networks is a growing field, it attracts many mathematicians, computer scientists and economic researchers. The research focuses on the approximation of continuous functions $C[a,b]$, by using neural networks.

In this research, we studied how multi-layered neural networks approximate functions with back-propagation aalgorithm.These neural networks are commonly referred to as multilayer perceptions (MLPS) which represent a generalization of the single layer perceptron.

The basic preliminaries of approximation are stated here. Uuniversal approximation theorem is an existence theorem in the sense that it provides the mathematical justification for the approximation of an arbitrary continuous function as opposed to exact representation. However, the theorem does not say that single hidden layer is optimum in the sense of learning time. The learning algorithm MLP improves the approximation as possible. Some continuous functions are approximated here, to reduce the approximation error by using MLP.

Contents

Nomenclature

Symbols	Definitions
$\|\|.\|\|$	Euclidean norm
$E_n(f)$	Degree of approximation
ε	Epsilon
d	Metric
$C[a,b]$	Continuous functions on [a,b]
x	Input
d	Target
w	Weights
y	Output
δ	Gradient
e	Error

Figures

Introduction

The approximation by neural networks attracts attentions of researchers, especially in the recent years. In all studies, the authors studied the degree of best neural approximation. The best approximation problem can be found in almost every book on approximation theory. This problem was studied from the second half of the19th century to the early 20th century, and by 1915 the main results had been established. In the 40^{th} of the last century, the study of neural networks begun to increase as a branch of artificial intelligence. The two topics (Approximation theory and neural networks) had been in touch in 1988 by Universal Approximation Theorem. In the beginning of the millennium, researchers became more interested in studying neural approximation. Many algorithms have been established since then. The algorithm of study has been applied to problems of optimizing the number of hidden neurons in a multilayer perceptron (MLP) and optimizing the number of learning epochs in MLP's back propagation training using both synthetic and benchmark data sets.

Our research consists of two chapters. The first chapter deals with a brief introduction to learning by neural approximation and what is function approximation. We also dealt with the best approximation to

the theorems of existence and uniqueness and some related definitions. Also, we talked about neural network, types of artificial neural network, mathematical model of neural networks which are explained through figures.It is of duty to clarify the relation between function aapproximations and neural networks through universal approximation theorem.

In chapter two, we deal with multilayer perceptron for function approximation; we gave an introduction to machine learning and multilayered perceptron neural network. Examples from several continuous functions are applied by the algorithm to show the ability of approximation by MLP. MSE is calculated to each example to have an exact eeror values.

Each theorem and algorithm of this work opens wide doors for new developments in the field of function approximation, neural network operators and activation functions, and training algorithms.

CHAPTER ONE

Brief Introduction to Learning by Neural Approximation

1.1. What is Function Approximation?[1]

Function approximation is a technique for estimating an unknown underlying function using historical or available observations from the domain. Before giving a specific definition of best function approximation, we need to know the following concepts

1.1.1. Normed Spaces [1]

By a normed space, mean a real vector space R^n in which every vector x is associated with a real number $\| x \|$. called its norm. That is.

for any vectors $x, y \in R^n$ and scalar a $\in R$, we have

1. $\|x\| \geq 0$.
2. $\|x\| = 0$ iff $x = 0$;
3. $\|ax\| = |a| \|x\|$; and
4. $\|x + y\| \leq \|x\| + \|y\|$ (Triangle inequality)

1.1.2. Definition of Metric Space[1]

Let$X$ be a set .A metric on X is an assignment of a distance $d(x, y) \in R$ to every pair of "points" x, y in X that is

$$d: X \times X \rightarrow R,$$

satisfying the following conditions:

1.(**Positivity**) For all $x, y \in X$, $d(x, y) \geq 0$ and

$d(x, y) = 0$ iff $x = y$,

2.(**Symmetry**) for all $x, y \in X$, $d(x, y) = d(y, x)$,

3.(**Triangle inequality**) for all $x. y. z \in$ X ,we have

$$d(x, y) + d(y, z) \geq d(x, z) .$$

A metric space is a set X together with such a metric d.

In every normed space. A norm induces a distance by the formula

$$\rho(x, y) = \|x - y\|$$

1.1.3 Best Approximation[2]

An approximation method requires a set of approximating functions, U say, which is a subset of X. Specifically, the method is just a mapping from X to U . In other words, given any $f \in X$, the method picks the element u , say, from U , which is regarded as an approximation to f . To find whether the method is good or bad, it should be compared with the best approximation.

1.1.4 Definition (Best Approximation).[1]

The best approximation of $f \in X$ from U is an element $u^* \in U$ s.t.

$$d(f, u^*) = \inf\{d(f, u): u \in U\} =: \text{dist}\{f, U\}$$

or

$$\|f - u^*\|_p = \inf_{u \in U} \|f - u\|_p.$$

1.1.5 Example (Polynomial Interpolation on $[a, b]$)[2]

Let X be $C[a, b]$ and let U be P_n, which are the linear spaces of continuous functions from $[a, b]$ to R and of polynomials of degree at most n, respectively. Then the following approximation method is useful. We pick points x_i, i = 0 .. n , that satisfy $a \leq x_0 < x_1 < \ldots < x_n \leq b$, and , given any $f \in C[a, b]$, we let $p = M(f)$ be the polynomial in p_n that satisfies the conditions .

$$P(x_i) = f(x_i), \qquad i = 0, \ldots, n$$

1.1.6 Definition (Degree of Approximation)[3]

Suppose we are given sequence (U_n) of subsets of X . Degree of approximation considers the behavior of

$$E_n(f) := dist\ (f, U_n)$$

as a function of the parameter $E_n(f) \to 0 (n \to \infty)$ first question whether our choice of approximation sets (U_n) is reasonable at all. The next question of interest is just how "fast" or "slow" this convergence is

far with Y ⊂ X, a class of function sharing with . particular F certain characteristics, one distinguishes

- Jachson – type (or , direct) theorems: $f \in Y \rightarrow E_n(f) = O(n^{-\propto})$
- Bernstein–type (or, inverse) theorems: $E_n(f) = O(n^{-\propto}) \rightarrow f \in Y$

1.2. Existence and Unicity of Best Approximations[4]

1.2.1.Existence Theorem

Let Y b a finite dimensional subspace of a normed linear space X , and let $x \in X$, there exist a (not necessarily unique) $y \in Y$ such that

$$\| x - y^* \| = \min_{y \in Y} \| x - y \| ,$$

for all $y \in Y$. That is , there is a best approximation to x by elements of Y.

1.2.2. Uniqueness Theorem[4]

In a strictly convex normed linear space a finite – dimensional subspace contains a unique point closest to any given point.

1.3 Neural networks[5]

Research among the sphere of neural networks has been attracting increasing attention in recent years. Since 1943, when Warren Meculloh and Walter Pitts [4] bestowed the primary model of artificial neurons, new and additional subtle proposals are made up of decade-to-decade. Mathematical analysis has solved a number of the mysteries expose by the new models. However, it has left several questions open for future investigation.

1.4 Types of Artificial Neural Networks[5]

1.4.1 Feedforward Neural Networks

The feedforward neural network was the priming and simplest kind. During this network, the knowledge moves solely from the input layer directly through any hidden layers to the output layer with no cycles loops, as in figure 1.1

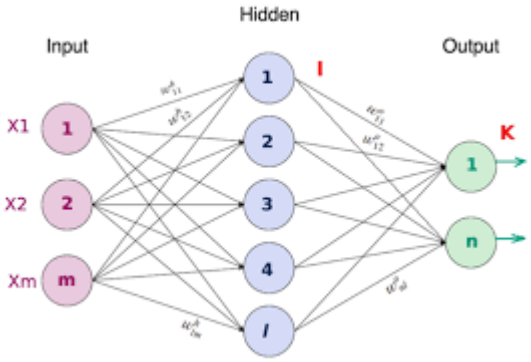

Fig. 1.1 Feedforward Neural Network

1.4.2 Regularity Feedback Networks

When the neural network has some kind of internal recurrence, meaning that the signals are feedback to a neuron or layer that has already received and processed that signal, the network is of the type feedback. See Figure 1.2.

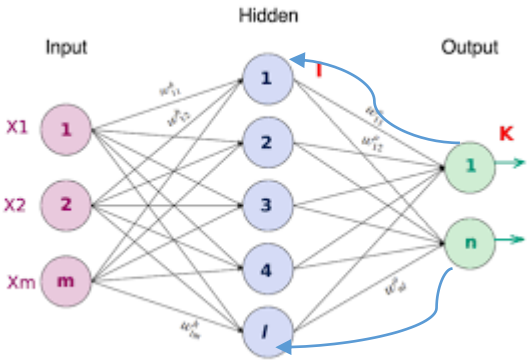

Fig.1.2 Feedback Neural Network

1.5 Artificial Neuron Networks (ANN)[7]

An artificial neuron is a function F_j of the input $x = (x_1, x_2, \dots, x_d)$ weighted by a vector of connection weights $w_j = (w_{j1}, \dots w_{jd})$ completed by a neuron bias b_j and associated to an activation function Ø, namely

$$Y_j = F_j(X) = Ø(\langle w_j . x \rangle + b_j)$$

1.6 Types of Neural Networks Activation Functions[7]

1-Binary step function

2-Linear activations

3-Non-linear activations functions.

There are many activation functions from the above types, each has advantages and disadvantages, we concern ourselves with only *tanh* activation function.

1.6.1 The Hyperbolic Tangent Function

In addition, neural networks with tanh activation function are much easier to learn and give a performance that is more effective. tanh is given by

$$\sigma(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

The curve of σ is shown in Fig1.1.

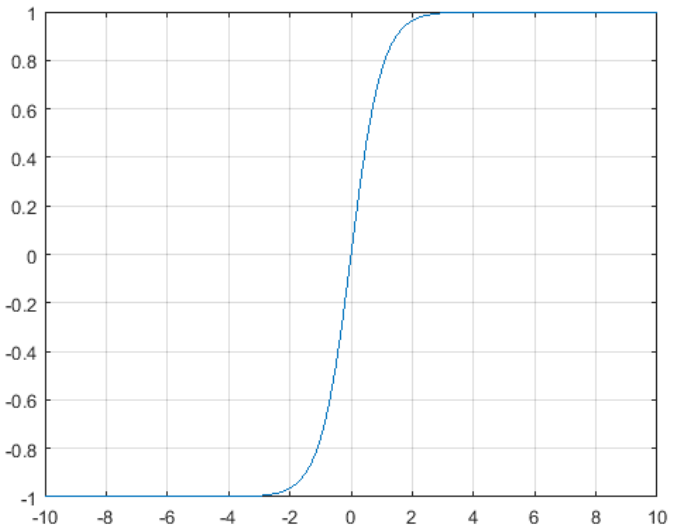

Fig.1.3 tanh Function

1.7 Mathematical Model of Artificial Neuron[5]

In the above, we have simply analyzed the structure and information – processing mechanism of the biological neuron, to provide biological bases for constricting the mathematical model of an artificial neuron obviously. It is impossible to simulate factually varicose characters of the biological neuron in a current computer, and we must make various reasonable simplification

In current research on the neural network, the neuron is the most essential information – processing unit of the neural network, generally, the mathematical model can be depicted as

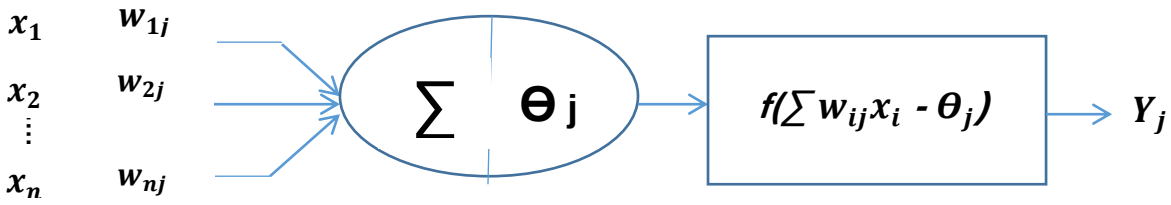

Fig.1.4 Artificial neuron model

In Fig1.2, $(x_i, i = 1, 2, \ldots . n)$ is the input signal of n external neurons to a neuron j; w_{ij} is the connection weight between the ith external neuron and the neuron j.The relationship

$$Y_i = f\left(\sum_{i=1}^{n} w_{ij}\ x_i - \Theta_j\right),$$

where f can be a non –linear activation function such as assign function or sigmoid function.

1.8 Neural Networks and Function Approximation[5]

A neural network can approximate any continuous function, provide it has at least one hidden layer and uses non-linear activations there. This

has been proven by the universal approximation theorem which is stated below.

1.8.1 Theorem UAT [3]

For any continuous function f defined on a compact space $X \subseteq R$, there exists a neural network of single hidden layer$N_n f$ that satisfies

$$\| f - N_n f \| < \in$$

CHAPTER TWO

Multilayer Perceptron for Function Approximation

Machine learning is a term arises from the field of artificial intelligence, but it has close parallels with approximation theory.In this chapter, we study multilayer feedforward networks, an important class of neural networks. Multilayer perceptron have been applied successfully to solve some difficult and diverse problems by training them in a supervised manner with a highly popular algorithm known as the error back-propagation algorithm. This algorithm is based on the error-correction learning rule.

2.1. Introduction to Machine Learning[7]

Machine learning is divided into two main types, supervised learning and unsupervised learning (there are other variations such as reinforcement learning but these are not relevant to the work covered in this research). Supervised learning is where a set of known (previously measured) samples are used to determine estimates of function.

Mitchell puts it that the “field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience”.

Artificial neural networks learn to approximate functions. In supervised learning, a dataset is comprised of inputs and outputs, and the supervised

learning algorithm learns how to best map examples of inputs to examples of outputs.

Neural networks are examples of a supervised learning algorithm and seek to approximate the function represented by your data. This is achieved by calculating the error between the predicted outputs and the expected outputs and minimizing this error during the testing process.

2.2. Multilayered Perceptron Neural Network[ ]

Typically, the multilayer feedforward network consists of a set of sensory units (source nodes) that constitute the input layer, one or more hidden layers of computation nodes, and an output layer of computation nodes. The input signal propagates through the network in a forward direction, on a layer-by-layer basis. These neural networks are commonly referred to as multilayer perceptrons (MLPS), which represent a generalization of the single layer perceptron considered in basically.

Error back-propagation learning consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, an activity pattern (input vector) is applied to the sensory nodes of the network, and its effect propagates through the

network layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of the networks are all fixed. During the backward pass, on the other hand, the synaptic weights are all adjusted in accordance with an error-correction rule. Specifically, the actual response of the network is subtracted from a desired (target) response to produce an error signal. This error signal is then propagated backward through the network, against the direction of synaptic connections hence the name "error back-propagation." The synaptic weights are adjusted to make the actual response of the network move closer to the desired response in a statistical sense.

The learning process performed with the algorithm is called back propagation learning. When training the system, the backward propagation will lead the system to reduce the error

2.3 Back-Propagation Algorithm[7]

The corresponding signal flow graph for back propagation learning, incorporating both the forward and backward phases of the computations involved in the learning process, is presented in Fig. 2.1The top part of the signal-flow graph accounts for the forward pass.

The lower part of the signal-flow graph accounts for the backward pass, which is referred to as a sensitivity graph for computing the local gradients in the back-propagation algorithm. Earlier we mentioned that the sequential updating of weights is the preferred method for online implementation of the back-propagation algorithm. For this mode of operation, the algorithm cycles through the training sample $\{(x(n), d(n))\}$ as follows:

1. **Initialization:** Assuming that no prior information is available, pick the synaptic weights and thresholds from a uniform distribution whose mean is zero and whose variance is chosen to make the standard deviation of the induced local fields of the neurons lie at the transition between the linear and saturated parts of the sigmoid activation function.
2. **Presentations of Training Examples.** Present the network with an epoch of training examples. For each example in the set, ordered in some fashion, perform the sequence of forward and backward computations described under points 3 and 4, respectively.
3. **Forward Computation.** Let a training example in the epoch be denoted by $(x(n), d(n))$,with the input vector $x(n)$applied to the

input layer of sensory nodes and the desired response vector $d(n)$presented to the output layer of computation nodes.

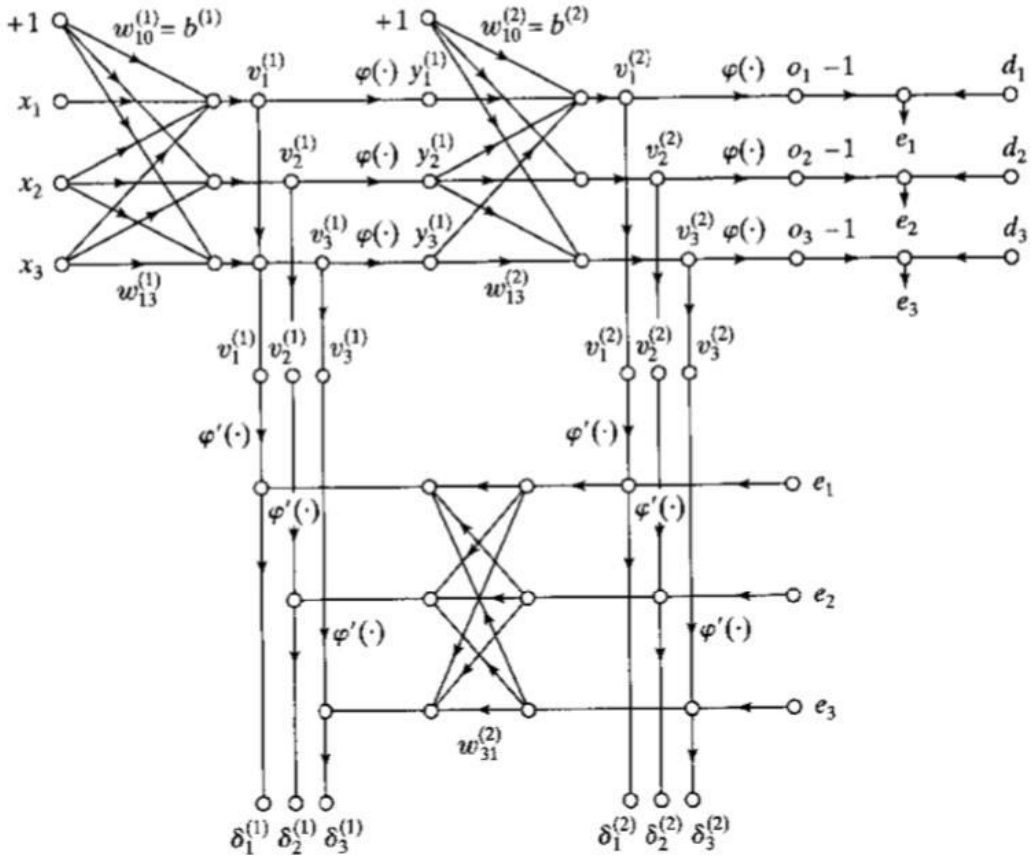

Fig 2.1. Back propagation Scheme

Compute the induced local fields and function signals of the network by proceeding forward through the network, layer by layer. The induced local field $v(n)$ for neuron j in layer $l$is

$$v_j^{(l)}(n) = \sum_{i=0}^{m0} w_{ij}^{(l)}(n) y_i^{(l-1)}(n),$$

where $y_i^{(l-1)}(n)$ is the output (function) signal of neuron $i$in the previous layer $l-1$ at iteration n and $w_{ij}^{(l)}(n)$ is the synaptic weight of neuron j in the layer l that is feed from neuron i in layer $l-1$.

- For $i = 0$, we have

$$y_0^{(l-1)}(n) = 1,$$

and

$$w_{j0}^{(l)}(n) = b_j^l(n)$$

is the bias applied to neuron j in the layer l. Assuming the use of sigmoid function, the output signal of neuron j in layer l is

$$y_j^{(l)} = \varphi_j(v_j(n))$$

- If neuron j is the first hidden layer (i.e., $l = 1$), set

$$y_j^{(0)}(n) = x_j(n),$$

where $x_j(n)$ is the jth element of the input vector $x(n)$.

- If neuron j is in the output layer(i.e., $l = L$, where L is referred to as the depth of the network), set

$$y_j^{(L)} = o_j(n)$$

- Compute the error signal

$$e_j(n) = d_j(n) - o_j(n),$$

where $d_j(n)$ is the jth element of the desired response vector $d(n)$.

4. **Backward computation.** Compute the δ (i.e., local gradients) of the network, defined by

$$\delta_j^{(l)}(n) = e_j^{(L)}(n)\varphi_j'\left(v_j^{(L)}(n)\right),$$

for neuron j in output layer L

$$\delta_j^{(l)}(n) = \varphi_j'\left(v_j^{(l)}(n)\right)\sum_k \delta_k^{(l+1)}(n)w_{kj}^{(l+1)}(n),$$

for neuron j in hidden layer l, where the prime in $\varphi_j'(.)$ denotes the differentiation with respect to the argument. Adjust the synaptic weights of the network in layer l according to the generalized delta rule:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha\left[w_{ji}^{(l)}(n-1)\right] + \eta\delta_j^{(l)}(n)y_j^{(l-1)}(n),$$

Where η is the learning-rate parameter and α is the momentum constant.

5. **Iteration.** Iterate the forward and backward computations under point 3and 4 by presenting new epochs of training examples to the network until the stopping criterion is met.

2.4 Approximations of Functions Related to Learning

A multilayer perceptron trained with the back propagation algorithm may be viewed as a practical vehicle for performing a nonlinear inputoutput mapping of a general nature. To be specific, let m_0, denote the number of input (source) nodes of a multilayer perceptron, and let $M = m_l$, denote the number of neurons in the output layer of the network. The input output relationship of the network defines a mapping from m_0 dimensional Euclidean input space to an M-dimensional Euclidean output space, which is infinitely continuously differentiable when the activation function is likewise. In assessing the capability of the multilayer perceptron from this viewpoint of input-output mapping, the following fundamental question arises: What is the minimum number of hidden layers in a multilayer perceptron with an input-output mapping that provides an approximate realization of any continuous mapping?

2.4.1 Universal Approximation Theorem[3]

The answer to the previous question is embodied in the universal approximation theorem for a nonlinear input-output mapping, which may be stated as:

Let$\emptyset(\cdot)$ be a non-constant bounded, and monotone-increasing continuous function. Let I_{m_0} denote the m_0- dimensional unit hypercube $[0,1]^{m_0}$ The space of continuous functions on I_{m_0} is denoted by $C(I_{m_0})$. Then, given any function $f \in C(I_{me})$ and $\epsilon > 0$, there exist an integer M and sets of real constants α_i, b_i and w_{ij}, where $j = 1, \ldots, m_0$ such that we may define

$$F(x_1, \ldots, x_{m_0}) = \sum_{i=1}^{m_1} \alpha_i \varphi \left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i \right) \qquad (2.1)$$

As an approximate realization of the function$f(.)$; that is,

$$\left| F(x_1, \ldots, x_{m_0}) - f(x_1, \ldots, x_{m_0}) \right| < \epsilon$$

for all $x_1, \ldots, x_{m_0}$ that lie in the input space.

The universal approximation theorem is directly applicable to multilayer perceptions . We first note that the tanh function

$$\frac{1}{[1 + exp(-v)]}$$

used as the nonlinearity in a neuronal model for the construction of a multilayer perceptron is indeed a non- constant, bounded, and monotone-increasing function; it therefore satisfies the conditions

imposed on the function $\emptyset(\cdot)$. Next, we note that (2.1) represents the output of a multilayer perceptron described as follows:

1. The network has m_0 input nodes and a single hidden layer consisting of m_0 neurons; the inputs are denoted by $x_1, \ldots, x_{m_0}$

2. Hidden neuron i has synaptic weights $w_{i_1}, \ldots, w_{im_0}$ and bias b_i.

3. The network output is a linear combination of the outputs of the hidden neurons, with $\alpha_1, \ldots, \alpha_{m_0}$, defining the synaptic weights of the output layer.

The universal approximation theorem is an existence theorem in the sense that it provides the mathematical justification for the approximation of an arbitrary continuous function as opposed to exact representation. In effect, the theorem states that a single hidden layer is sufficient for a multilayer perceptron to compute uniform approximation to a given training set represented by the set of inputs $x_1, \ldots, x_{m_0}$, and a desired (target) output $f(x_1, \ldots, x_{m_0})$. However, the theorem does not say that a single hidden layer is optimum in the sense of learning time, ease of implementation, or (more importantly) generalization. For that, many approximation theorems by MLP networks have been studied, beginning with Barron in his paper [7].

2.5 Summary of Back – Propagation Algorithm

1- Initialize weights with with *M*=0 AND V=1

3- Run iteration for all nodes *j* in the MLP

$$v_j^{(l)}(n) = \sum_{i=0}^{m0} w_{ij}^{(l)}(n) y_i^{(l-1)}(n),$$

4- Compute the error signal

$$e_j(n) = d_j(n) - y_j(n),$$

5- If $e_j(n)$ is accepted, then $y_j(n)$ is approved, else

5-1- Back computation

$$\delta_j^{(l)}(n) = \varphi_j'\left(v_j^{(l)}(n)\right) \sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n),$$

5-2- Adjust weight i of the node j with

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha\left[w_{ji}^{(l)}(n-1)\right] + \eta\delta_j^{(l)}(n) y_j^{(l-1)}(n),$$

6- Repeat 5

7- Compute MSE

2.6 Bounds on Approximation Errors[7]

Barron (1993)[6] has established the approximation properties of a multilayer perceptron, assuming that the network has a single layer of hidden neurons using sigmoid functions and a linear output neuron. The network is trained using the back-propagation algorithm and then tested with new data. During training, the network learns specific points of a target function f in accordance with the training data, and thereby produces the approximating function F defined in Eq. (2.1)When the network is exposed to test data that have not been seen before, the network function F acts as an "estimator" of new points of the target function.

2.6.1.Definition of Mean Square Error(MSE)[5]

MSE is the measure of how well the function approximation .The optimization problem is to minimize MSE as to reach to the target error. MSE is given by

$$\|x - Y\| = \left(\sum_{i=1}^{d} (x_i - y_i)^2 \right)^{1/2},$$

The universal approximation theorem is important from a theoretical viewpoint, because it provides the necessary mathematical tool for the viability of feedforward networks with a single hidden layer as a class of

approximate solutions. Without such a theorem, we could conceivably be searching for a solution that cannot exist. However, the theorem is not constructive, that is, it does not actually specify how to determine a multilayer perceptron with the stated approximation properties.

The universal approximation theorem assumes that the continuous function to be approximated is given and that a hidden layer of unlimited size is available for the approximation. Both of these assumptions are violated in most practical applications of multilayer perceptron.

The problem with multilayer perceptions using a single hidden layer is that the neurons therein tend to interact with each other globally. In complex situations this interaction makes it difficult to improve the approximation at one point without worsening it at some other point. On the other hand, with two hidden layers the approximation (curve-fitting) process becomes more manageable. In particular, we may proceed as follows :-

I. Local features are extracted in the first hidden layer. Specifically, some neurons in the first hidden layer are used to partition the input space into regions, and other neurons in that layer learn the local features characterizing those regions.

2. Global features are extracted in the second hidden layer. Specifically, a neuron in the second hidden layer combines the outputs of neurons in the first hidden layer operating on a particular region of the input space, and thereby learns the global features for that region and outputs zero elsewhere.

2.7 Approximation of Continuous Functions

In this section, we apply MLP networks with back-propagation to approximate some functions from continuous spaces.

2.7.1. Periodic Function

Let f be a real valued periodic function which is given by

$$f(x, y) = \sin(2x)\sin(2y)$$

Figure 2.2. shows the function f and its approximated neural network

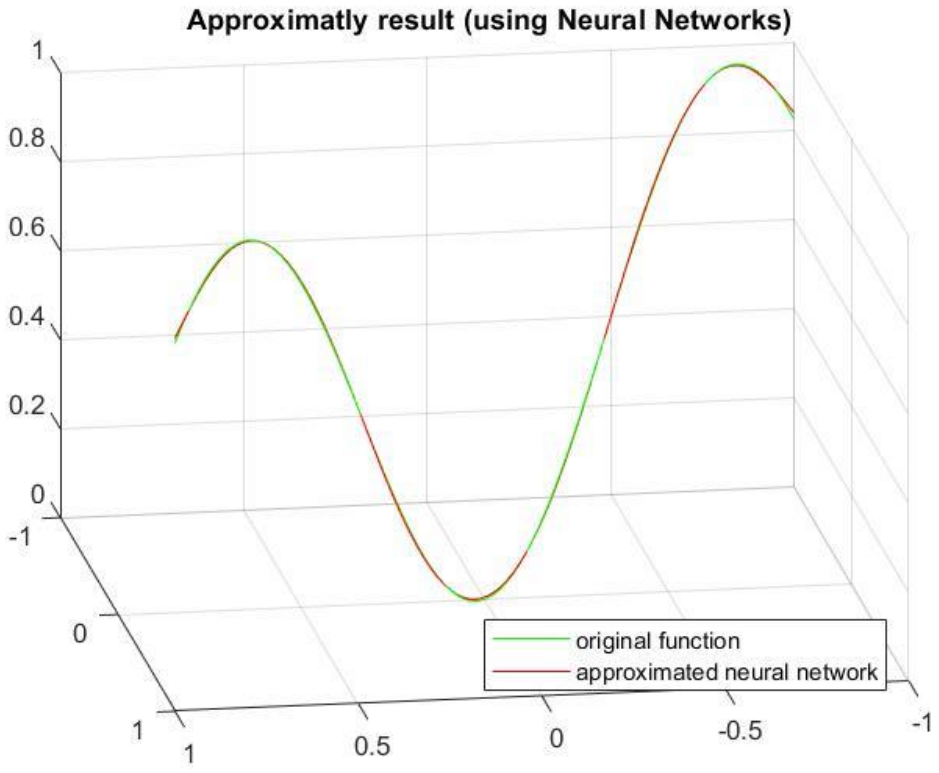

Fig.2.2. Approximation of periodic function2.7.1

Figure 2.3 shows the reduction of the error through 1000 epochs.

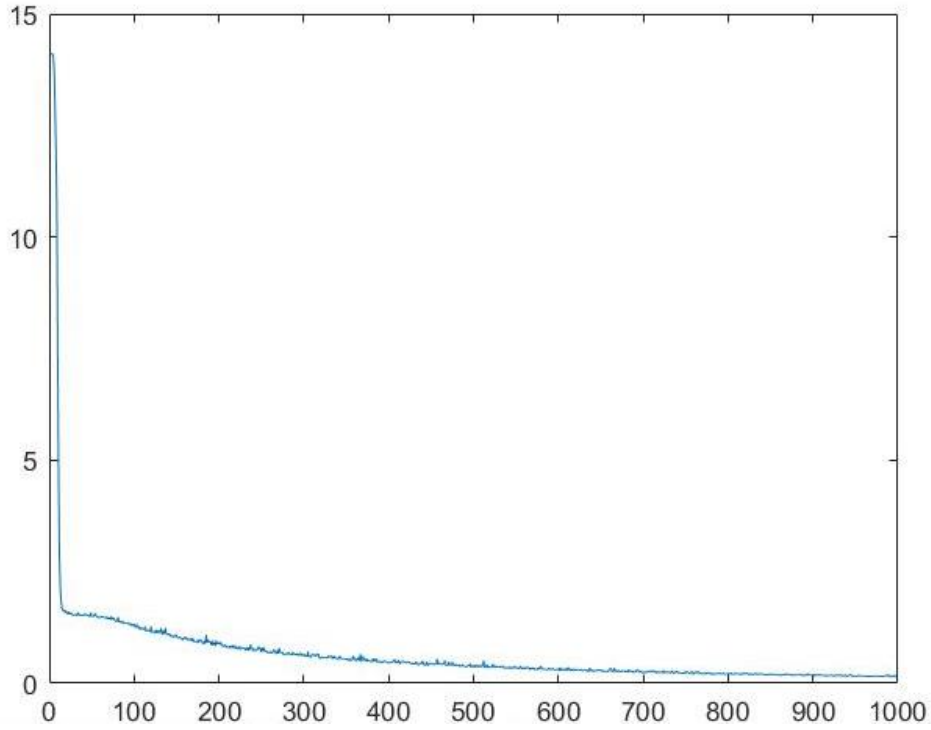

Fig.2.3. Error of periodic function 2.7.1

2.7.2 Lim Function

We apply MLP to approximate Lim function which is always used for approximation from [ ] $x, y \in$ [0,1]

$$f(x,y) = \frac{1}{6}\,[30 + 5xsin(5x)(4 + e^{-5y})]$$

The curves of the function and its approximation are graphed in the figure below,

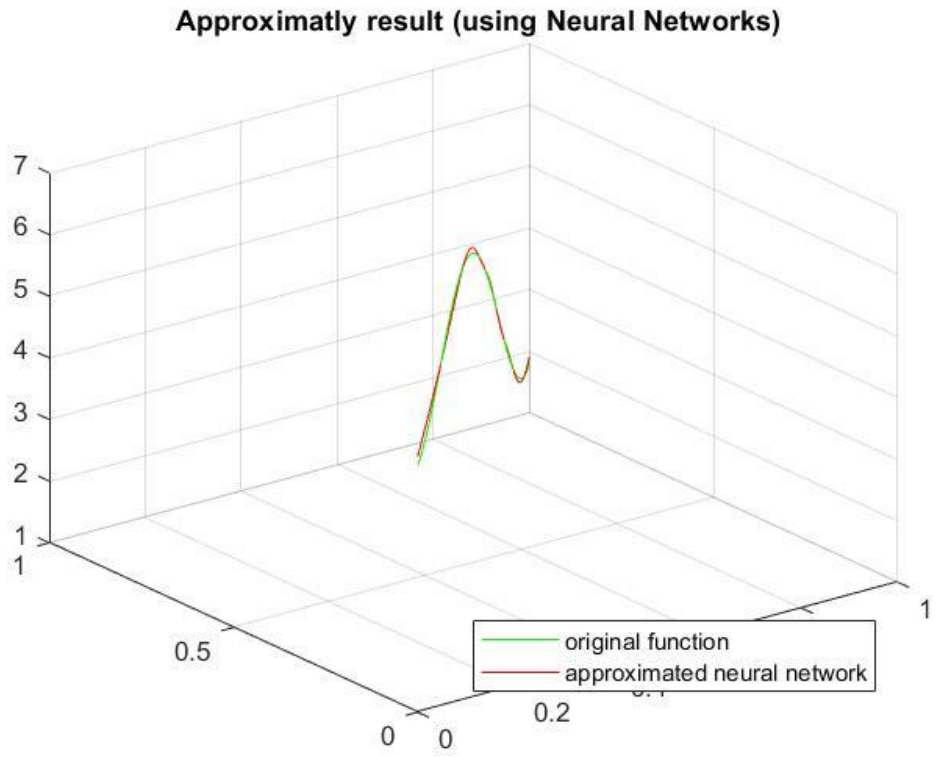

Fig.2.4. Approximation of periodic function 2.7.2

The error is given in the figure below, we notice that it converges to zero as the algorithm runs.

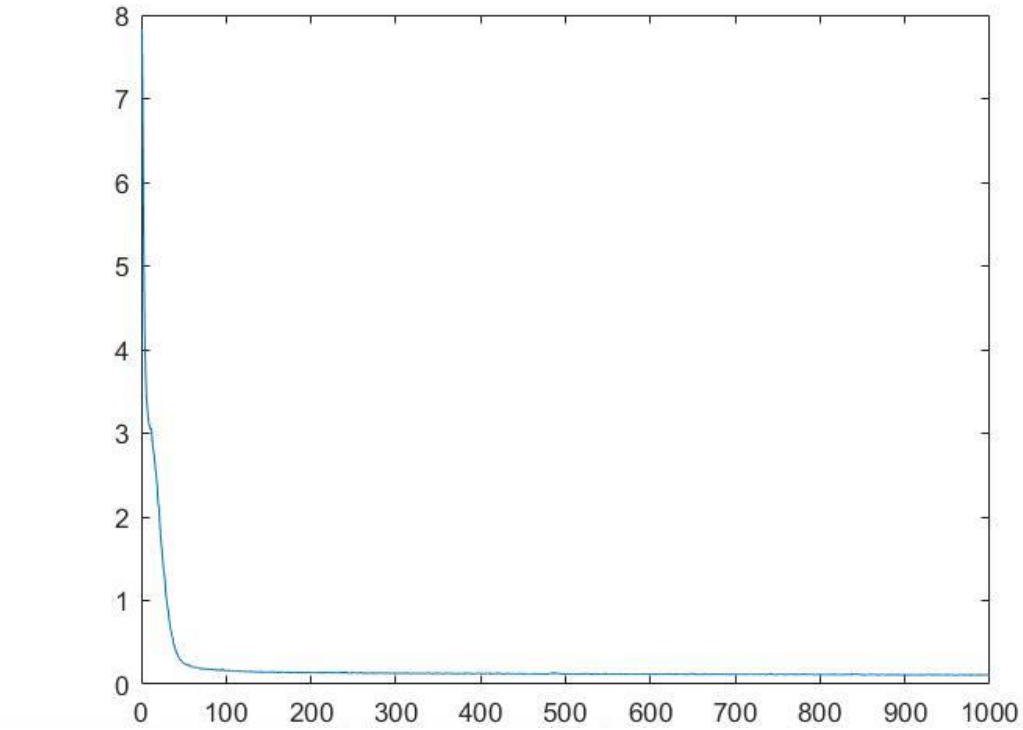

Fig.2.5. Approximation error of periodic function 2.7.2

2.7.3 ExponentialFunction

Another function that is used for approximation, it is given by

$$f(x,y) = 2sin(\pi e^{-x2-y2})$$

The following figure shows both the target function and the approximated one using MLP. Moreover, the error is calculated in the next figure.

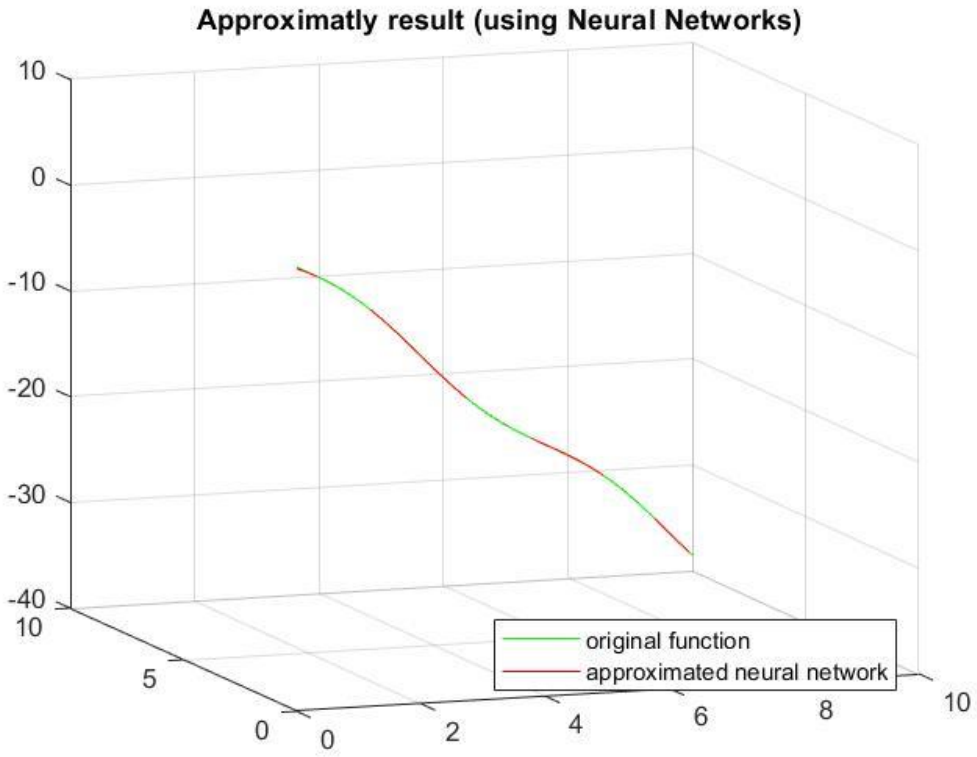

Fig.2.6. Approximation of exponential function2.7.3

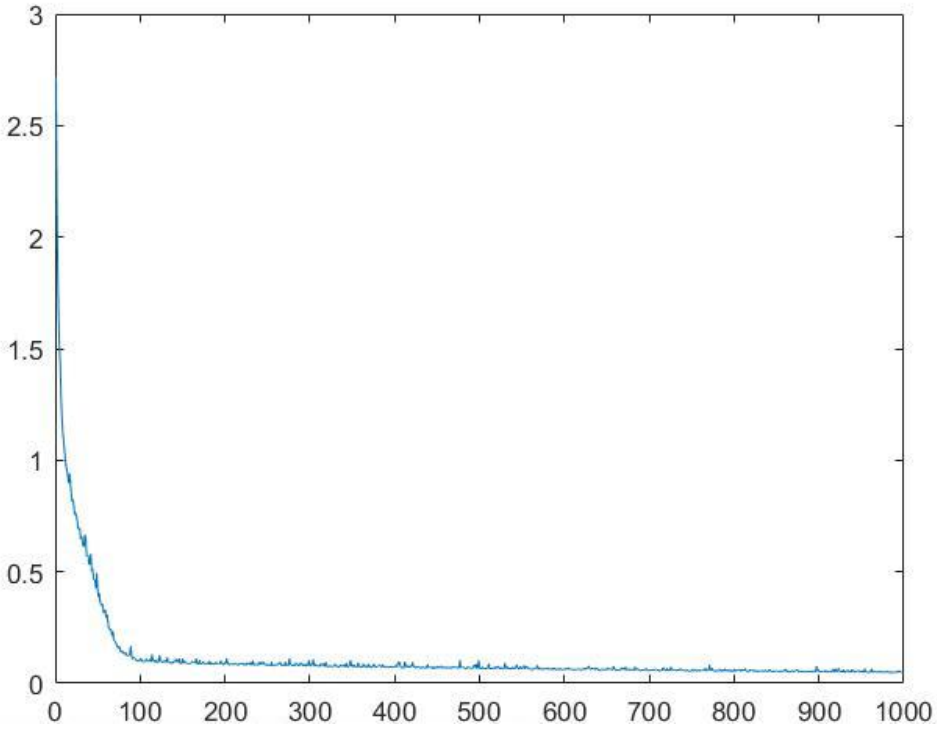

Fig.2.7. Approximation error of exponential function 2.7.3

2.7.4. MSE

We calculate the MSE for the above three examples. The following table shows the values for each one,

Table 2.1. MSE

Function	MSE
$\boldsymbol{f(x,y) = \sin(2x)\sin(2y)}$	1.3391e-04
$\boldsymbol{f(x,y) = \frac{1}{6}\,[30 + 5xsin(5x)\left(4 + e^{-5y}\right)]}$	2.6566e-04
$\boldsymbol{f(x,y) = 2sin(\pi e^{-x2-y2})}$	4.1827e-06

Conclusions

References

[1] N.L. Carothers, Approximation Theory. Bowling Green State University,1998.

[2]H. A. ALmurieb, & E. S. Bhaya, "The Degree of Best Neural Networks Approximation with Application," PhD Thesis, University of Babylon,2020.

[3] J.K. Steffens, ”The history of Approximation theory. From Euler to Bernstein- Birkhauser, 2006.

[4] E.W.Cheney, & E.D.Wulbert, "The Existence and Unicity of Best Approximation," Mathematical Scandinavica, vol.24, 1969, https://doi.org/10.7146/math.scand.a-10925.

[5] IBM Cloud Education, "Neural Networks", 2020, https://www.ibm.com/cloud/learn/neural-networks

[6] S. Haykin, "Neural Networks a Comprehensive Foundation," Pearson Prentice Hall,Canada,1994.

[7] J. P. Bridge, "Machine learning and Automated Theorem Proving," Technical Report, vol.792, pp.37-38, 2010.

[8] Z. Zainuddin & O. Pauline, "Function Approximation Using Artificial Neural Networks," International Journal of Systems Applications, Engineering& Development, vol.4, pp.176-177, 2007

المستخلص

يعتبر التقريب بواسطة الشبكات العصبية مجالًا علمياً متناميًا، يجذب العديد من الباحثين في الرياضيات والكومبيوتر والعلوم والاقتصاد.

باستخدام الشبكات العصبية ذات الطبقة المخفية، يمثل تطوير خوارزمية الانتشار العكسي علامة بارزه في الشبكات العصبية من حيث انه يوفر طريقه فعاله من الناحية الحسابية لتدريب الشبكات متعددة الطبقات على الرغم من إننا لا نستطيع الادعاء بأن خوارزمية الانتشار العكسي تؤمن الحل الأمثل لجميع المشكلات القابلة للحل.

حيث يشار الى هذه الشبكات العصبية عادة باسم الشبكات متعدد الطبقات والتي تمثل تعميماً للشبكة أحادية الطبقة وقد تم تطبيقها بنجاح لحل بعض المشكلات الصعبة والمتنوعة من خلال تدريبهم بطريقه خاضعة للأشراف باستخدام خوارزمية الانتشار العكسي للخطأ.

تعتمد هذه الخوارزمية على قاعدة تعلم تصحيح الأخطاء . يتم ضبط الأوزان المشبكية وفقًا لقاعدة تصحيح الخطأ على وجه التحديد ،فيتم طرح الاستجابة الفعلية للشبكة من الاستجابة (الهدف) المرغوبة لإنتاج إشارة خطأ ، ثم يتم نشر إشارة الخطأ هذه للخلف عبر الشبكة ضد اتجاه الوصلات المشبكية. ومن هنا جاء أسم "خطأ الانتشار الخلفي" من الآن فصاعداً سوف نشير إليها على أنها خوارزمية الانتشار العكسي وتسمى عملية التعلم التي يتم أجرائها باستخدام خوارزمية تعلم الانتشار العكسي.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية التربية للعلوم الصرفة

تقريب الدوال بواسطة خوارزمية MLP

بحث مُقدمّ

إلى مجلس كلية التربية للعلوم الصرفة في جامعة بابل كجزء من متطلبات نيل درجة الدبلوم العالي تربية/ الرياضيات

من قبل:

أحلام محمد عبد حمزة

بإشراف

د. حوراء عباس المرعب

١٤٤٢هـ ٢٠٢١ م