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 GRNN Algorithm

A project

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

BY:

Anwar Anwer Hamody kazem

Supervised by:

DR. HAWRAA ABBAS FADHIL ALMURIEB

2021 A.D ***1443 A.H***

بسم الله الرحمن الرحيم

﴿وَقُل رَّبِّ زِدْنِي عِلْماً﴾

سورة طه - أية 114

صدق الله العظيم

DEDICATION

Dedicate My project to

MY FATHER AND MY MOTHER

AND ESPECIALLY MY HUSBAND *Zaid*

MY BROTHERS **ALI, MOSTAFA, YASIR, MOHAMED ALI**

MY SISTERS

EATHEL,FATIMA,WUROOD,ZAHRAA,ALAA,ZAINAB

WHO ALWAYS ENCOURAGED AND SUPPORTED ME, WHOSE SUPPORT MADE THIS WORK POSSIBLE.

ACKNOWLEDGEMENT

Foremost, I would thank My God who helped me and saved me from failure. I would like to thank the person who guided me to the right path and worked very hard with my Research, Dr Hawraa A Almurieb. I extend my sincere thanks to the department, the college and all my loyal professors.

I would like to give special thanks to my family for their

great support and enthusiasm that helped me to accomplish this work. This research is dedicated to my husband Zaid, whose unwavering support has made this thesis possible.

Finally, I am grateful to all who offered help during the

the accomplishment of this work.

Anwar Anwer Hamody

ABSTRACT

Function approximation by neural networks still gets the interest of researchers for its significant usage in different fields. One of the demands is that approximation methods represent physical reality as accurately as possible .In our research ,we define a new neural network operater carrying properties from "the old" bernstien polynomials in 1921 . We get good rate of approximation in terms of modulus of continuity .

However, in its earlier results, the approximated neural networks may go far away from the target. So the training role arises here to change the values inside the network to be closer to the target. Our work applies a training algorithm called generalized regression neural network (GRNN), recently used for function approximation. It has a radial basis layer and a special linear layer.We conclude that it gives good numerical results by reducing the difference between any continuous function on the interval [0,1] and the approximated one resulting by GRNN. Mean square error (MSE) is used to calculate the value of that difference error accurately.A table of the results are there to judge the MSE It is great to add more panties related to what we presented in this research,

not only limited to the field of function approximation but also to employ the results in various practical studies.

CONTENTS

List of figures

NOMENCLATURE

Symbols	Explanation
C [a, b]	The space of all continuous functions defined on [a, b]
$(X, \|\| \cdot \|\|)$	**Normed space**
$\|\|\cdot\|\|$	**Euclidean Norm**
R^2	**The set of real numbers of dimension 2.**
$B_r(x)$	**ball of centre (1,0) and small radius r**
R	**The Space of Real Numbers**
$N_n(x)$	**Neural Network**
$c_{j,k}$	**Constants**
ε	**Epsilon**
$C^d[0,1]$	**The space of all continuous functions** **defined on [0,1]^d**
$B_n(f)$	**Bernstain Polynomial of Degree n**
X	**Inputs**
W	**Weights**
$Y(x)$	**Output**
ρ^2	**Variance**
$N_{n,d}(f)$	**Neural Network**
$E_n(f)_\infty$	**Degree of Best Uniform Approximation**
$\omega_f(x)$	**Modulus of smoothness**

ABBREVIATIONS

symbols	*Explanation*
Pes	*Processing Elements*
ANNs	*Artificial Neural Networks*
GRNN	*General Regression Neural Networks*
MLP	*Multilayer Perceptron*
BR	*Bayesian Regularization*
RBFN	*Radial Basis Function Networks*
MSE	*Mean Square Error*
RBF	*Radial Basis Function*
FFNNs	*Feedforward Neural Networks*

INTRODUCTION

Function Approximation is concerned with how the function is best approximated with simpler functions (Polynomials or Splines) or more applicable operators (Neural Networks or Wavelets).

Moreover, the term neural networks mean global function approaches that represent almost any function. Networks are seen as more powerful than simple M Models. So, it is better to think of feedforward Networks as function approximation.

In this research a new neural network operates is presented and , we use the learning abilities that neural networks have to approximate functions from the space $C[0,1]$. The Mean Square Error is evaluated to show how close the approximate is.

In chapter one, principles of approximation theory have been studied, especially for the branch of function approximation. The main theorems of approximation, such as the Existence and Uniqueness theorems, are stated. Also, we introduce the basics of neural networks that are related to function Approximation. This includes biological and mathematical expression of ANNs, their architectures and learning from function data.Our main result is to

present an existence theorem by defining a new neural network and estimate its degree of approximation .

In Chapter Two, we explain how training neural networks are used to best approximate functions. Many algorithms can find approximations by a neural network. General Regression Neural Networks (GRNN) is a good choice for this purpose. We use the algorithm to approximate several functions as examples of the quality of GRNN. We measure this quality by computing the Mean Square Error (MSE) between the target function and the approximated one.

More functions from higher dimensions can be approximated using GRNN or other training algorithms. More comparisons among algorithms can be discussed

CHAPTER ONE

Introduction To Neural Approximation

Rapidly during the last century, neural approximation has been increasing in importance many fields, such as numerical analysis, statistics, engineering, computer science, and others.

1.1. Function Approximation[1]

The theory of approximation is a very extensive field, which has various applications. The problem of function approximation had been studied throughout polynomials (trigonometric and algebraic), spline, wavelets and neural networks. In this section, we introduce fundamental ideas and aspects of approximation theory in the normed space X. Approximation theory is concerned with how functions can best be approximated with simpler functions (polynomials or splines) and quantitively characterize the errors introduced there. More generally, one may want to set up the practically helpful criterion for the quality of approximations. Given a set X of functions to be approximated and a set Y of functions by which the element of X are approximated, one may consider the problems of existence, uniqueness, and construction of a "best approximation" in the sense of such a criterion.

A natural setting for the problem of approximation is as follows

Definition 1.1.1[2]

Let $X = (X, \|.\|)$ be a normed space and suppose that any given $x \in X$ is approximated by a $y \in Y$, where Y is a fixed subspace of X. we let δdenoted the Euclidean distance from x to y, then

$$\delta(x, Y) = inf_{y \in Y} \|x - y\|$$

Clearly, δ depends on both on x and Y If there exists a $y_0 \epsilon Y$ such that

$$\|x - y_0\| = \delta$$

then $y_0$is called a best approximation to x out of Y.

We see that the best approximation y_0 is an element of minimum distance from the given x to the space Y such a $y_0 \in Y$ may or may not exist, and this raises the problem of existence. Then we have the following

1.1.2. Existence Theorem [2]

If Y is a finite-dimensional subspace of a normed space $X = (X, \|.\|)$, then for each $x \in X$, there exists a best approximation to x out of Y. The uniqueness problem is of practical interest, too, since for given

x and Y there may be more than one best approximation, as we shall see. For obtaining uniqueness of best approximation, this suggests the following

1.1.3 Definition[3]

A norm $\|\cdot\|$ on a vector space, X is said to be strictly convex if for any

$x \neq y \in Y$, with

$$\|x\| = r = \|y\|,$$

we always have

$$\|\lambda x + (1-\lambda)y\| < r,$$

for any $0 < \lambda < 1$.

Geometrically, the open line segment between any pair on the ball's surface of radius r in X lies entirely inside the ball.

To arrive at a somewhat easier condition to check, let's translate our origin definition into a statement about the triangle inequality in X.

1.1.4. Lemma[3]

X has a strictly convex norm if and only if the triangle inequality is strict on non−parallel vectors, that is, if and only if

$$\text{for all } x \neq\propto y, y \neq\propto x \propto\in R$$

we have,

$$\|x + y\| < \|x\| + \|y\|.$$

We may summarize the uniqueness result as follows

1.1.5. Uniqueness Theorem[2]

In a strictly convex normed space, there is at most one best approximation to an $x \in X$ out of a given subspace Y. To take a more precise look at those problems, we give the following examples, including several existence cases and the best approximation's uniqueness.

1.1.6. Example[1]

Consider $X = R^2$ under the norm

$$\|(x_1, x_2)\|_2 := (|x_1|^2 + |x_2|^2)^{1/2},$$

and consider the subspace

$$Y = \{(0, y) \mid y \in R\} \text{ (the Y} - \text{axis)}$$

Existence and uniqueness of the best Approximation to the point $x = (1,0)$ out of Y are guaranteed by theorems (1.1.2) and (1.1.5), respectively since Y is 1 −dimensional subspace of the strictly convex space under the norm$\|.\|_2$

To find the best approximation of $x = (1,0)$, we draw a ball of centre $(1,0)$ and small radius r, which is

$$B_r(x) = \{y \in X: \|x - y\|_2 < r\}$$

in $(X, \|.\|_2)$, then we make the radius bigger and bigger to intersect Y, the unique intersection point here is $(0,0)$, which is the individual best approximation to $x = (1,0)$ out of Y, as shown in Figure(1.1)

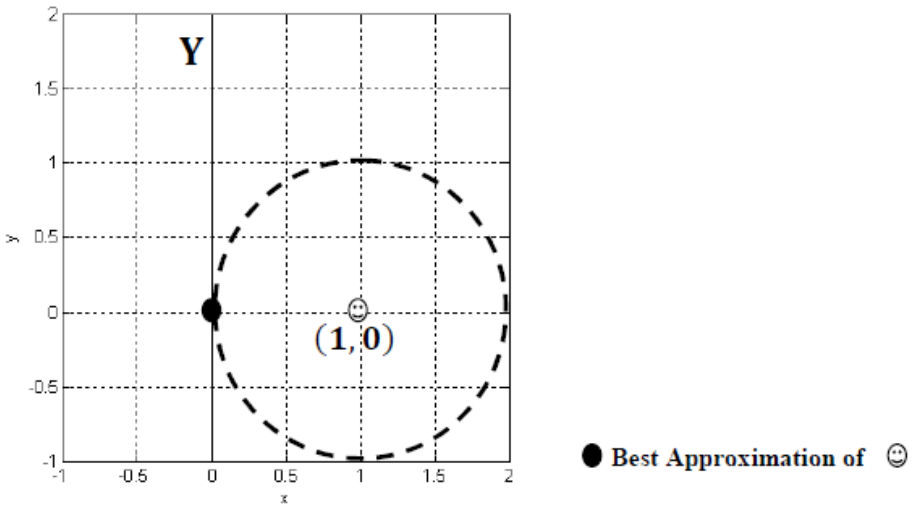

Figure1.1. The best approximation to x out of Y in the norm space $(R^2 , \|.\|_2)$

1.2. Introduction to Neural Networks[14]

The human nervous system contains billions of cells, which are referred to as neurons. A biological neuron is composed of multiple dendrites, a nucleus and an axon. The neurons are connected using axons and dendrites, and the connecting regions between axons and dendrites are referred to as synapses. These connections are illustrated in the Figure1.2.

The strengths of synaptic connections often change in response to external stimuli. This change is how learning takes place in living organisms. This biological mechanism is simulated in artificial

neural networks, which contain computation units referred to as neurons.

When a stimuli is sent to the brain, it is received through the synapse located at the extremity of the dendrite. When a stimuli arrives at the brain, it is transmitted to the neuron via the synaptic receptors, which adjust the strength of the signal sent to the nucleus. This message is transported by the dendrites to the nucleus to be then processed in combination with other signals emanating from other receptors on the other dendrites. Thus the combination of all these signals takes place in the nucleus. After processing all these signals, the nucleus will emit an output signal through its single axon. The axon will then stream this signal to several other downstream neurons via its axon terminations. Thus a neuron analysis is pushed in the subsequent layers of neurons.

On the other hand, artificial neural networks are built on the principle of bio-mimicry. External stimuli (the data), whose signal strength is adjusted by the neuronal weights (remember the synapse?, circulates to the neuron (a place where the mathematical calculation will happen) via the dendrites. The result of the calculation – called the output – is then re-transmitted (via the axon)

to several other neurons, and then subsequent layers are combined, and so on.

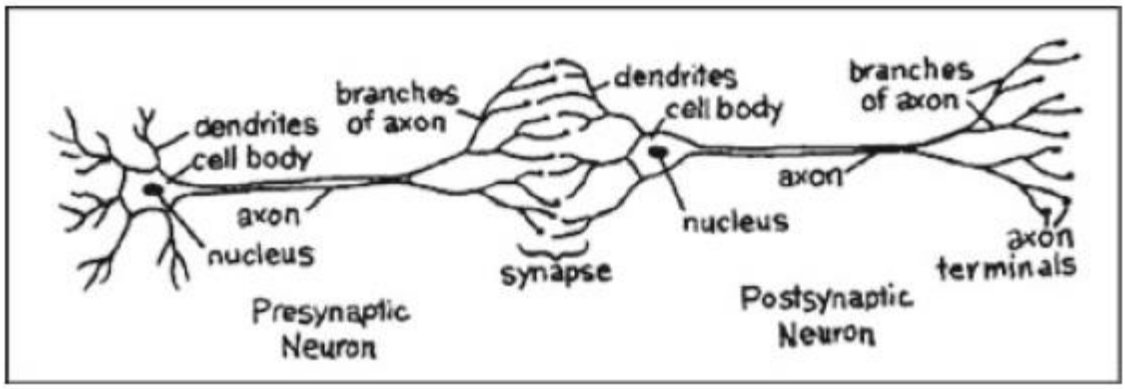

Figure.1.2. The Synaptic connections between neurons.

Artificial neural networks are popular machine learning techniques that simulate the mechanism of learning in biological organisms.

1.3. Mathematical Formula of Neural Networks[1]

The general common mathematical expression among all studies is given by

$$N(x) = \sum_{i=1}^{n} c_i \, \sigma(w_i x_i + \theta_i) \qquad (1.3)$$

σ:- activation function

w_i:-weights.

c_i:- Constants.

x_i:-inputs.

θ_i:- thresholds.

Miscellaneous models of neural networks are born frequently. Each model has a certain structure, characters, approximation abilities, and processing methods. According to some dimensions, many types of neural networks are classified, we list some of them in the following section

1.4. Neural Network Architectures [5]

ANNs can be viewed as weighted directed graphs in which nodes are artificial neurons (PEs), and directed edges (with weights) are connections from the outputs of neurons to the inputs of neurons. Based on the connection pattern (architecture), ANNs can be grouped into two significant categories Feedforward networks in which no loop exists in the graph and feedback networks in which loops exist because of feedback. The most common family of feedforward networks is the layered neural network, in which neurons are organized into layers with connections strictly in one direction from one layer to another. All the networks with no loops

can be rearranged in layered feedforward networks with possible skip-layer connections. Figure (1.3) also shows typical networks of each category connections.

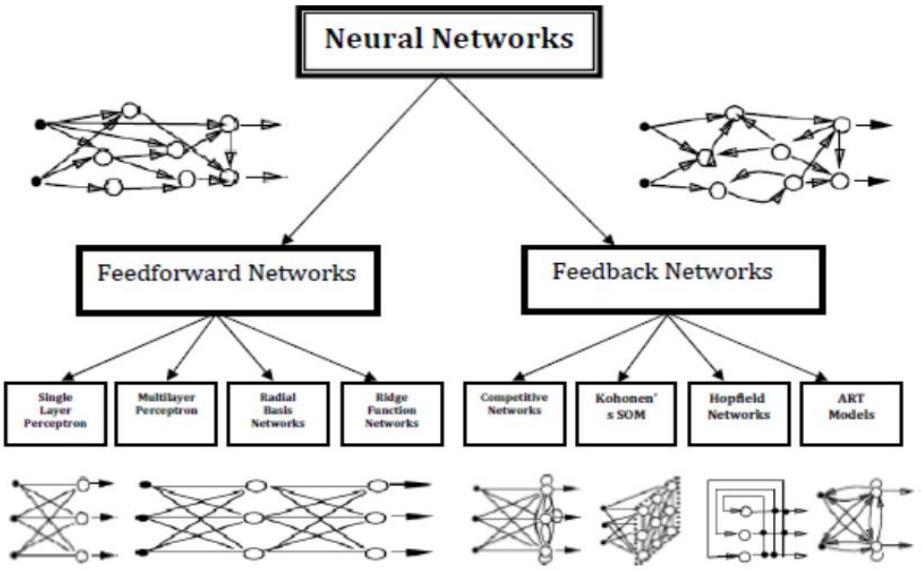

Figure.1.3. A Taxonomy of network architectures.

1.5. Machine Learning with Neural Networks[6]

Neural networks are parameterized models that are learned with continuous optimization methods. Neural networks can be viewed as more powerful versions of these simple models, with this power being achieved by combining the basic models into a comprehensive neural architecture.

On the other hand, as the amount of data increases, neural networks have an advantage because they retain the flexibility to model more complex functions with the addition of neurons to the computational graph.

Moreover, the ability to put the basic units together in a clever way is a key architectural skill required by practitioners in deep learning. Nevertheless, it is also essential to learn the properties of the basic models in machine learning since they are used repeatedly in deep learning as elementary units of computation; therefore, explore these basic models.

1.5.1. Deep Learning [7]

Deep learning is a subset of machine learning in artificial intelligence with networks capable of learning unsupervised from unstructured or unlabeled data. Also known as deep neural learning or deep neural network. Deep learning is an *AI* function that mimics the workings of the human brain in processing data foto detect objects, recognize speech, translate languages, and make decisions.

Deep learning *AI* is able to learn without human supervision, drawing from data that is both unstructured and unlabeled.

The problem of learning a mapping between an input and an output space is equivalent to synthesizing an associative memory that retrieves the appropriate output when presented with the input and generalizes when shown with new inputs.
It is also equivalent to the problem of estimating the system that transforms inputs into outputs given a set of examples of input-output pairs. *A* classical framework for this problem is approximation theory.

1.6. Neural Network and Approximation Theory[17]

Learning an input-output mapping from a set of examples of the type that many neural networks have been constructed to perform can be regarded as synthesizing an approximation of a multi-dimensional function that solves hypersurface reconstruction. From this point of view, this form of learning is closely related to classical approximation techniques,

Approximation theory deals with the problem of approximating or interpolating a continuous, multivariate function $f(X)$ by an approximating function $F(W; X)$ having a fixed number of parameters x and W are real vectors $X = x_{1,}x_{2,}\ldots,x_d$ and $W = (w_1, w_2, \ldots\ldots, w_d)$.

For a choice of a specific F , the problem is then to find the set of parameters W that provides the best possible approximation of f on the set of examples. This is the learning step.

Needless to say, it is essential to choose an approximating function F that can represent f as well as possible. There would be little point in trying to learn if the selected approximation function F $(W; X)$ could only give a very poor representation of $f(X)$, even with optimal parameter values. Therefore, it is helpful to separate three main problems:

1) the problem of which approximation to use, i.e. which classes of functions $f(X)$ can be effectively approximated by which approximating functions $F(W; X)$. This is a representation problem.

2) the problem of which algorithm to use for finding the optimal values of the parameters W for a given choice of F.

3) the problem of an efficient implementation of the algorithm in parallel, possibly analogue hardware.

Almost all approximation schemes can be mapped into some network that can be dubbed as a neural network.

Networks of this type, with one layer of hidden units, can approximate uniformly any continuous deviate functions (see, for instance, Cybenko [9] or Funahashi [10]. They proved similar results about approximating functions by neural networks but with different neural details.

1.6.1. Universal Approximation Theorem[5]

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

$$\|f - N_n f\| < \varepsilon$$

In general,

each approximation scheme has some specific algorithm for finding the optimal set of parameters W. In the next chapter, we deal with an algorithm that weight the neural network to have the best approximation of functions from C[a,b].

1.7. Construction of Neural Networks

The aim of this chapter is to define a new neural network coperator with a new activation function to approximat functions from the space $C^d[-1,1]$ lay them .The space $C^d[-1,1]$ consists of continuous functions of the form

$$f(\boldsymbol{x}) = \big(f(x_1), f(x_2), \dots., f(x_d)\big)$$

Our activation functions of jth component is given by

$$\sigma_j(\boldsymbol{x}) = x_j^k \big(1_j - x_j\big)^{n-k},$$

where $j = 1, \dots., d,\ k = 0, \dots., n,$ and $x_j, 1_j \in [-1,1],$ are the jth component of $\mathbf{1}$=$(1_j, \dots., 1_d)$ and $\boldsymbol{x} = (x_1, \dots., x_d), \mathbf{1}, \boldsymbol{x} \in [-1,1]^d$

The neural network with our activation function is given by

$$N_n(\boldsymbol{x}) = \sum_{j=0}^{d} \sum_{k=0}^{n} c_{j,k}\ \sigma_j(\langle \boldsymbol{w}.\boldsymbol{x} \rangle + b_j)\ ,$$

where

$$c_{j,k}\ = \frac{n!}{k!\,(n-k)!} f\left(\frac{k}{n}\right).$$

$$\sigma_j(\boldsymbol{x}) = x_j^k\left(1_j - x_j\right)^{n-k},$$

1.7.1. Best Neural Approximation

Theorem I (Existence Theorem)

Let $f \in C^d[0,1]$, then for any $n \in N$ with $d \leq n$ there exist a neural network

$$N_{n,d}(f) = \sum_{j=1}^{d}\sum_{k=0}^{n} c_{j,k}\,\sigma_j(\langle \boldsymbol{w}.\boldsymbol{x}\rangle + b_j)\,,$$

such that

$$E_n(f)_\infty \leq c\;\omega_f(n^{-\frac{1}{2}})$$

Where

$$c_{j,k} = \frac{n!}{k!\,(n-k)!}\,f\left(\frac{k}{n}\right)$$

$$\sigma_j(\boldsymbol{x}) = x_j^k\left(1_j - x_j\right)^{n-k},$$

with $j = 1, \ldots . d\,, k = 0, \ldots , n\,, 1_j$ and x_j are the jth component of $\mathbf{1} = (1_1, \ldots , 1_d)$ and $\boldsymbol{x} = (x_1, \ldots . , x_d), \mathbf{1}, \boldsymbol{x} \in [-1,1]^d$.

As we have seen in 1.1.2 and 1.4.1, approximation theorems were proven in terms of ε. If one wants to improve the estimates , then

modulelus of continuity is there to measure ε, so it gives better degree of approximation than ε .

To learn more about modulus of continuity, here is the following definition

1.7.2.Modulus of Continuity [12]

For any bounded function f on $[a, b]$,the modulus of continuity of f is given by

$$\omega_f([a,b];\delta) = \sup\{|f(x) - f(y)| : x, y \in [a,b], |x-y| \leq \delta\}$$

From above , we note that ω_f is a measure of ε that goes along with δ in the definition of uniform continuity .

1.7.3. Properties of ω_f [12]

1- If f is uniformly continuous then $\omega_f(\delta) \longmapsto 0 \ \ as\ \delta \longmapsto 0^+$

2- If $\delta` < \delta, than\ \omega_f(\delta`) \leq \omega_f(\delta)$

3- $\omega_f(n\delta) \leq n\ \omega_f(\delta), for\ n \epsilon N$

4- $\omega_f(\lambda\delta) \leq (1+\lambda)\omega_f(\delta), for\ \lambda > 0$

Now, we benefit from the following known Bernstein's polynomials to prove our main result,

1.7.4. Bernstein's Polynomials[12]

Bernstein's polynomials were defined in 1912 by

$$\big(B_n(f)\big)(x) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right)\binom{n}{k} x^k (1-x)^{n-k}$$

for $0 \leq x \leq 1$

In general,

1. $\sum_{k=0}^{n} \binom{n}{k} x^k \, (1-x)^{n-k} \; = 1$

2. $\sum_{k=0}^{n} \left(\frac{k}{n} - x\right)^2 \binom{n}{k} x^k (1-x)^{n-k} \leq \frac{1}{4n}$

3. For $\delta > 0, 0 \leq x \leq 1, \left|\frac{k}{n} - x\right| \geq \delta$, implies

$$\sum_{k=0}^{n} \binom{n}{k} x^k (1-k)^{n-k} \leq \frac{1}{4n\sigma}$$

The tools to prove our main result are ready.So here is our existence theorem,

1.7.5. Proof of theorem I

Let $n \in \boldsymbol{N}$ *and* $d \leq n$, *then by lemma* 1.1.4 *and* 1.7.2,

$$|f - N_{n,d}(f)| = \left| f(y) - \sum_j \sum_k c_{j,k}\ \sigma_j(y) \right|$$

$$= \left| \sum_k \sum_j f(y_j) - f\left(\frac{k}{n}\right) \frac{n!}{k!\,(n-k)!} {y_j}^k (1-y_j)^{n-k} \right|$$

$$\leq \sum_k \sum_j \left| f(y_j) - f\left(\frac{k}{n}\right) \frac{n!}{k!(n-k)!} {y_j}^k (1-y_j)^{n-k} \right| \text{ by holder inequality}$$

$$\leq \sum_k \sum_j \left(\left| \left(f(y_j) - f\left(\frac{k}{n}\right) \right) \frac{n!}{k!\,(n-k)!} {y_j}^k (1-y_j)^{n-k} \right| \right)$$

$$\leq \sum_k \sum_j \omega_f \left(\left| y_j - \frac{k}{n} \right| \right) \frac{n!}{k!\,(n-k)!} {y_j}^k (1-y_j)^{n-k}$$

By using $\lambda = \sqrt{n} \left| y - \frac{k}{n} \right|$ and $\delta = \frac{1}{\sqrt{n}}$ *in property*(3) *of* 1.7.3 then

we use Cauchy-shwarz inequality,we get

$$|f - N_{n,d}(f)|$$

$$\leq \omega_f\left(\frac{1}{\sqrt{n}}\right) \sum_k \sum_j \left[1 + \sqrt{n} \left| y_j - \frac{k}{n} \right| \right] \frac{n!}{k!\,(n-k)!} {y^k}_j (1-y_j)^{n-k}$$

$$= \omega_f\left(\frac{1}{\sqrt{n}}\right) \left[1 + \sqrt{n} \sum_k \sum_j \left| y_j - \frac{k}{n} \right| \frac{n!}{k!\,(n-k)!} {y^k}_j (1-y_j)^{n-k} \right]$$

$$\leq \omega_f\left(\frac{1}{\sqrt{n}}\right)\left[+\sqrt{n}\sum_j\left\{\sum_k\left|y_j-\frac{k}{n}\right|\frac{n!}{k!\,(n-k)!}y_j^k\ (1\right.\right.$$

$$\left.\left.-y_j)^{n-k}\right\}^{\frac{1}{2}}\left\{\sum_k\left|y_j-\frac{k}{n}\right|\frac{n!}{k!\,(n-k)!}y_j^k\ (1-y_j)^{n-k}\right\}^{1/2}\right]$$

$$\leq \omega_f\left(\frac{1}{\sqrt{n}}\right)\left[1+\sqrt{n}\sum_j^d\frac{1}{4n}\right]^{\frac{1}{2}}$$

$$\leq c\ \omega_f\left(\frac{1}{\sqrt{n}}\right)\blacksquare$$

CHAPTER TWO

Training Neural Networks to

Best Approximate Functions

"It is best to think of feedforward networks as function approximation machines that are designed to achieve statistical generalization, occasionally drawing some insights from what we know about the brain, rather than as models of brain function. ..." [4]

2.1. Training Artificial Neural Networks[1]

All neurons of a given layer are generating output, but they don't have the same weight for the next neurons layer. This means that if a neuron on a layer observes a given pattern, it might mean less for the overall picture and will be partially or completely muted. This is what we call weighting: a significant weight means that the input is essential, and of course, a small weight means that we should ignore it. Every neural connection between neurons will have an associated weight. And this is the magic of neural network adaptability: weights will be adjusted over the training to fit our objectives. In simple terms: training a neural network means finding the appropriate weights of the neural connections.

Many algorithms conclude neural networks to best approximate functions. Linear Regression, General Regression Neural Networks (GRNN), Multilayer Perceptron (MLP), Radial Basis Function Networks (RBFN), and Bayesian Regularization (BR) are some of the significant used algorithms to that purpose, as well as other purposes that use approximation techniques.

This chapter uses the GRNN algorithm to find a neural approximation for some different continuous functions to train that neural network to get closer to the target function. Mean Square Error (MSE) is the measure of how well the function approximation is. The optimization problem is to minimize MSE as to reach the target error.

2.2.GRNN(General Regression Neural Networks)[11]

Algorithms are considered one of the essential things in our world today because many human inventions are based on them. Algorithms are the reason why computers perform mathematical operations. For this reason, Al-Khwarizmi is considered the godfather of computers while not forgetting the role of John von Neumann. Today, you can search billions of files in a matter of seconds, and you can also calculate all bank commissions in seconds, which is due to the great progress in the science of algorithms.

RBF and GRNN methods are efficient to approximate continuous functions. From the approximation perspective, a neural network can be the approximate unknown model or problem and scape. Therefore, any function can be represented by the weighted sum of a

group of basic functions. In GRNN, this is possible via the transfer function of neurons in hidden layers. During the function approximation process using RBF and GRNN, the dispersion constant should be the same as the resolution of functions, which is the distance between the input vectors. Large and small dispersion constants will lead to fewer and more neurons. Fewer neurons will cause the overlap of the inputs and outputs. Sub sequent, the neural network cannot provide different responses, and the approximation will be in non-fit condition.

On the contrary, more neurons will result in the over-fit phenomenon in the approximation. From the results, RBF and GRNN are better approached when it comes to function approximation.

GRNN is a variation to radial basis neural networks. GRNN was suggested by D.F. Specht in 1991[11]. GRNN can be used for regression, prediction, and classification. GRNN can also be a good solution or an online dynamical system.GRNN represents an improved technique in the neural networks based on the nonparametric regression. The idea is that every training sample will represent a mean to a radial basis neuron.[2]

2.2.1. Implementation[3] [4]

GRNN has been implemented in many computer languages, including MATLAB,[3] R-programming language and Python. Neural networks (specifically Multi-layer Perceptron) can delineate non-linear patterns in data by combining with generalized linear models by considering the distribution of outcomes (sightly different from original GRNN) Linear Regression has been generalized by [12]. GRNN is an FNN with a radial basis layer and the next linear layer. It generates a good tool for function approximation. It is simply an input-output with a structure of Radial Basis Network in the first layer, but with a different input to the transfer function. The Euclidean distance is applied to the input X and the weight W as follow

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

where $X = (x_1, x_2, \cdots, x_d)$ and $W = (w_1, w_2, \cdots, w_d)$. Also, the bias b allows the sensitivity of the rabbi's neuron to be adjusted. The radial basis neural network is simply like the following figure shows

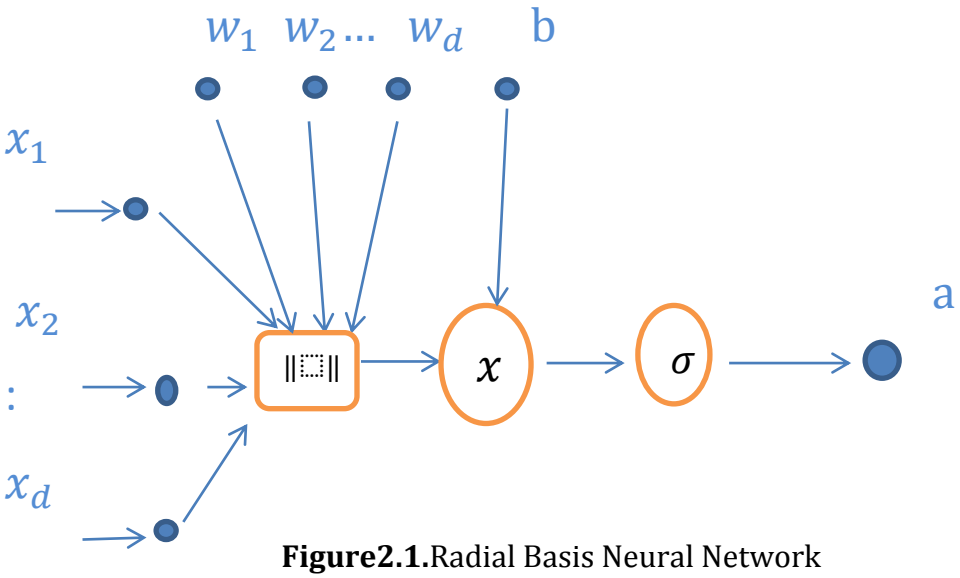

Figure2.1. Radial Basis Neural Network

Moreover, the process of using GRNN in the FFNNs is shown in the following figure,

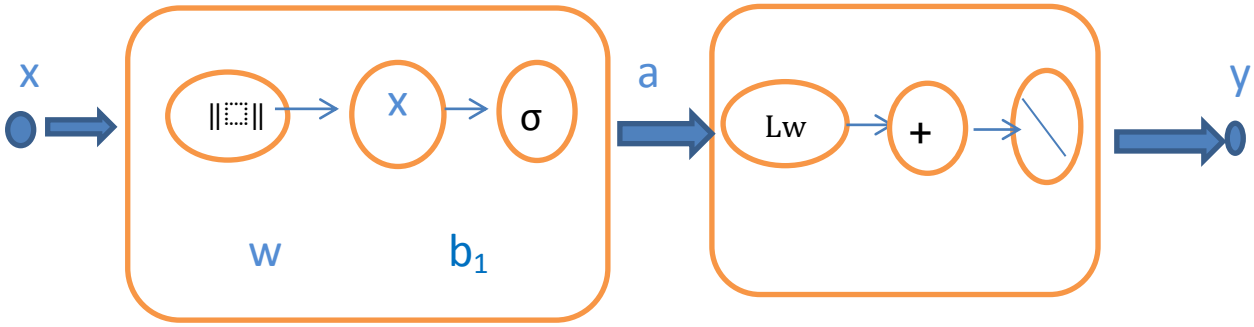

Figure 2.2. Structure of GRNN

And then, the result is activated by Gaussian RBF as follow

$$\sigma(x_i) = e^{-{x_i}^2/\rho^2}$$

The final neural network is as follow

$$Y(\boldsymbol{x}) = \frac{\sum_{i=1}^{d} Y(x_i) e^{-(x_i - w_i)^2/\rho^2}}{\sum_{i=1}^{d} e^{-(x_i - w_i)^2/\rho^2}},$$

where ρ^2 is the smoothing parameter of the Gaussian kernel (Default 1). $Y(\boldsymbol{x})$ represents the probability density function of the normal distribution.

Theoretically, RBF maps an input X to get a weight W, so we could measure how much the function is the approach to the target by calculating $\|\boldsymbol{X} - \boldsymbol{Y}\|$. Thus, the result is trained again in the same procedure.[5]

2.3. Experimental Results

We introduce some practical results for our algorithm, including function examples, and one of the most important neural network algorithms for function approximation is the GRNN algorithm. Matlab programs it. In addition, all calculations and figures are done

by Matlab too. In this section, we apply the algorithm to several examples, including

2.3.1. Continuous Exponential Function

We apply GRNN to approximate a continuous exponential function which is of one variable $x \in [0,1]$

$$\boldsymbol{y = (x + 1)\ exp\ (-3x + 3)}$$

The following figure shows both the target function and the approximated one using GRNN, and the MSE is 0.0042

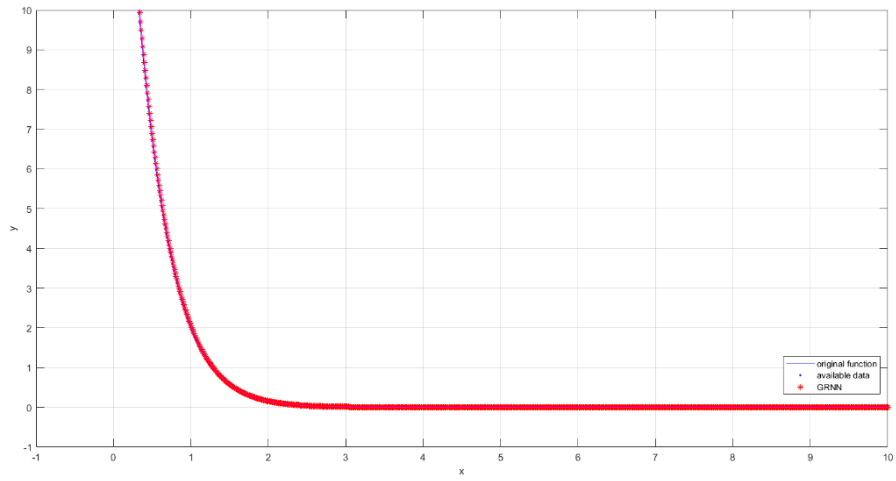

Figure2.3. Approximation of $\boldsymbol{y = (x + 1)\ exp\ (-3x + 3)}$ by GRNN

2.3.2 Continuous Periodic Function

We apply GRNN to approximate the function of a continuous periodic function, which is of one variable $x \in [0,1]$

$$\boldsymbol{y = sin(4x)exp(-|5x|)}$$

The following figure shows both the target function and the approximated one using GRNN; the MSE is 0.0031

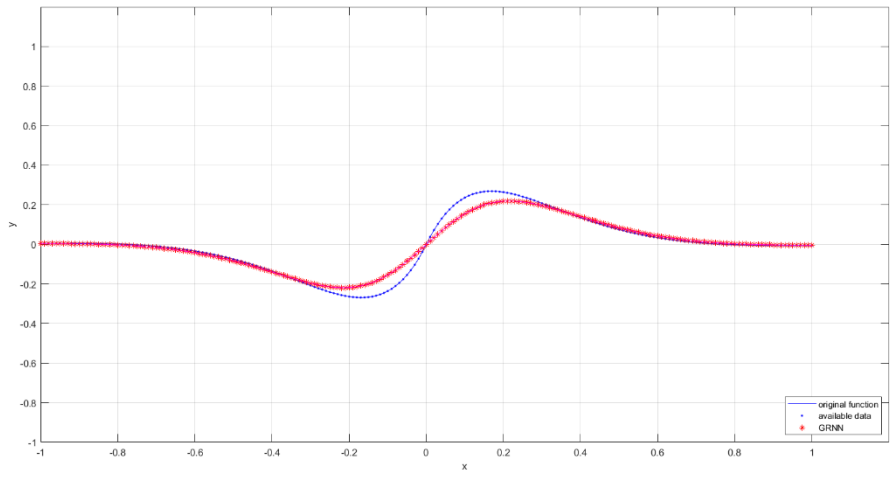

Figure.2.4. Approximation of $\boldsymbol{y = sin(4x)exp(-|5x|)}$ by GRNN

2.3.3 Santner Function

Another famous function that is used for approximation is the Lim Function [5]. It is given by

$$y = f(x) = e^{-1.4x}\cos(3.5\pi x)\,, x \in [0,1]$$

The following figure shows both the target function and the approximated one using GRNN; the MSE is 1.4637e-05

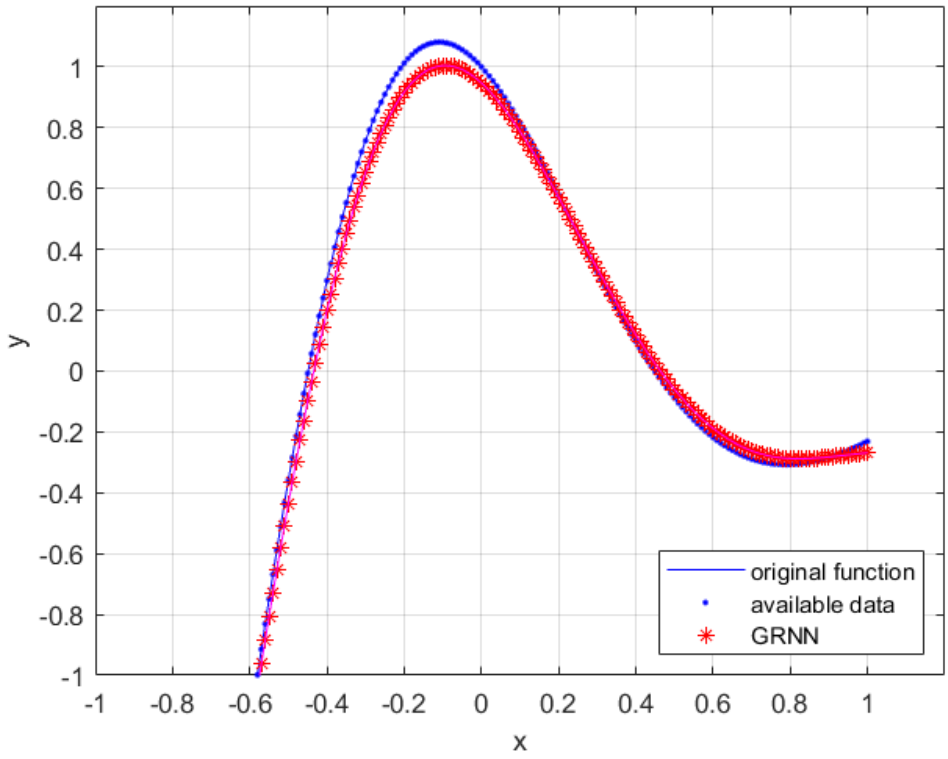

Figure.2.5. Approximation of $f(x) = e^{-1.4x}\cos(3.5\pi x)$ by GRNN

2.3.4. MSE Results

Finally, we collect our results of MSE of each functions in the examples above with the following table,

FUNCTION	MSE
$\boldsymbol{y = (x+1)exp(-3x+3)}$	0.0042
$\boldsymbol{y = sin(4x)exp(-\lvert 5x\rvert)}$	0.0031

$\boldsymbol{y = e^{-1.4x} cos(3.5\pi x)}$	1.4637e-05

Conclusions of future work

Neural networks centre around the theory of function approximation. Among the parameters used to approximate functions, neural networks occupy a large portion of them in various practical fields.

On the other hand, neural networks are considered universal approximates. So we can look at neural networks in view of function approximation.

From now on, more research exploration continues for more useful neural networks to approximate functions well and to find algorithms that serve these tasks.

The neural network will be applied to a wider range of benchmark functions to validate its efficiency in future work.

References

[1] G. Cybenko, “Continuous Valued Neural Networks: Approximation Theoretic Results,” in *Computer Science and Statistics: proceedings of the 20th Symposium on the Interface,* 1988, pp. 174–183.

[2] A. Dingankar, D. Phatak, and B. County, “Simultaneous Approximation with Neural Networks,” no. June 2014, 2000.

[3] C. Ding, F. Cao, and Z. Xu, “The Essential Approximation Order for Neural Networks with Trigonometric Hidden Layer Units,” *Springer-Verlag Berlin Heidelberg*, vol. 3971, pp. 72–79, 2006.

[4] E. S. Bhaya, “Neural Network Trigonometric Approximation,” *Journal of Babylon University/Pure and Applied Sciences*, vol. 24, no. 9, pp. 2395–2399, 2016.

[5] D. Costarelli and G. Vinti, “Max-product Neural Network and Quasi-Interpolation Operators Activated by Sigmoidal Functions,” *Journal of Approximation Theory*, vol. 209, no. October 2017, pp. 1–22, 2016.

[6] H. A. Almurieb, “Simultaneous Approximation of Order m by Artificial Neural Network,” vol. 56, no. 4, 2017.

[7] S. Sonoda and N. Murata, “Neural Network with Unbounded Activation Functions is Universal Approximator,” 2017.

[8] C. K. Chui, S.-B. Lin, and D.-X. Zhou, “Construction of Neural Networks for Realization of Localized Deep Learning,” *Frontiers in Applied Mathematics and Statistics*, vol. 4, pp. 1–22, 2018.

[9] S. Dittmer, E. J., and P. Maass, “Singular Values for ReLU Layers,” *IEEE Transactions on Neural Networks and Learning Systems*, no. 2, pp. 1–12, 2019.

[10] S. Ibrahim, N. A. Wahab, F. S. Ismail, and Y. M. Sam, “Optimization of Artificial Neural Network Topology for Membrane Bioreactor Filtration Using Response Surface Methodology,” *IAES International Journal of Artificial Intelligence*, vol. 9, no. 1, pp. 117–125, 2020.

[11] H. Almurieb and E.Bhaya, "the Degree of the best Neural Networks:Approximation with Application",2020phD thesis ,University of Babylon.

[12] N. L. Carothers, *A Short Course on Approximation Theory*. Citeseer, 1998.

[13] E. S. Bhaya and H. A. Almurieb, “On the Shape Preserving Approximation by Spline Polynomials,” MSc Thesis, University of Babylon, 2010.

[14] G. Anastassiou," Intelligent Systems: Approximation by Artificial

Neural Networks "University of Memphis, 2011 Springer- Verlag Berlin Heidelberg ,pp.10-116.

[15] A.K.Jain, J.Mao, K.M.Mohiddin, "Artificial Neural Networks: A Tutorial", Computer, 1996, pp.31-44.

[16] C. Aggarwal "Neural Networks and Deep Learning" Springer International Publishing AG, part of Springer Nature 2018 PP.21_512.

[17] X. He and Sh. Xu "Process Neural Networks Theory and Applications", Springer-Verlag Berlin Heidelberg April _2009, PP.14_253.

[18] G. Cybenko, "Approximation by Superpositionsof a Sigmoidal Function," *Math. Control. Signals Syst.*, vol. 2, no. 4,

[19] Funahashi On the Approximate Relization of Continuous Mappings by Neural , "Neural Networks ,vol.2,no.3,pp.183-192,1989.

[20] D. F. Specht, "A General Regression Neural Network," *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576, 1991.

[21] Z. Zainuddin and O. Pauline "Function Approximation Using Artificial Neural Networks" 2007, vol. 1, no 4, pp. 173_178.

المستخلص

لا يزال موضوع تقريب الدوال عن طريق الشبكات العصبية يحظى باهتمام الباحثين، نظراً لاستخدامه الرائع في مختلف المجالات. وإن أحد أهم المتطلبات هو أن تمثل طريقة التقريب الواقع المادي بأكبر قدر ممكن من الدقة في هذا البحث قمنا بتعريف مشغل الشبكه العصبيه الذي يحمل خصائص من بيرنشتاين القديم من حيث الحدود في عام 1912 وحصلنا على معدل جيد لمعامل التقريب للاستمراريه ومع ذلك ، فإن الشبكات العصبية التقريبية ، في نتائجها الأولية ، قد تذهب بعيدًا عن الهدف. لذلك ينشأ دور التدريب هنا لتغيير القيم داخل الشبكة لتكون أقرب إلى الهدف المنشود. في هذه المساهمة المتواضعة ، سوف نطبق خوارزمية لتقريب الدوال تسمى بخوارزمية GRNN ، والتي تمتلك طبقة أساس شعاعي وطبقة خطية خاصة ، وقد أعطت نتائج جيده لتقريب الدوال وتقليص قيمة الفرق بين الدالة الاصلية ونظيرتها المقربة. هذا وقد تم استخدام قيمة الخطأ التربيعي ، والذي يرمز له بالرمز MSE ، لقياس دقة الخطأ الناتج عن هذه الخوارزمية.

إنه لأمر رائع أن نضيف المزيد من الأعمال المستقبلية المتعلقة بما قدمناه في هذا البحث ، ليس فقط في مجال تقريب الدوال، ولكن أيضًا لتوظيف النتائج في دراسات عملية متنوعة.

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة بابل

كلية التربية للعلوم الصرفة

قسم الرياضيات

تقريب الدالة بواسطة خوارزمية GRNN

مشروع

مقدم الى مجلس التربية للعلوم الصرفة جامعة بابل

كجزء من متطلبات نيل درجة الدبلوم العالي تربية/الرياضيات

من قبل

انوار انور حمودي كاظم

بأشراف

م .د. حوراء عباس فاضل المرعب