Modelling and Solving Differential Equations using Neural Networks: A Study

R. Devipriya
Research scholar, Periyar University
Salem, Tamilnadu
devi.jpn@gmail.com

S. Selvi
Assistant Professor
NKR Government Arts College(W)
Namakkal, Tamilnadu
sselvimaths@yahoo.com

***Abstract*- Generally, Neural Networks (NN) are considered as a hierarchical models that can be used to learn patterns or knowledge from data with complicated nature or distribution. These NNs are also used as universal function approximators. Therefore, NNs can be applied to solve the mathematical problems, as numerical analysis tool. This paper discusses applications of neural networks in modelling and finding solution of various differential equations.**

Keywords - Modelling and Solving Differential Equations, Differential Equations, Neural Network.

1. INTRODUCTION

Over the last few decades Neural Networks has showed considerable significance and attention due to their application in various disciplines such as electro-chemistry, visco elasticity, optics, star cluster etc. Since Neural networks are best at identifying patterns or trends in data, they are well apt for predicting needs including sales forecasting, data validation, risk management, target marketing, under sea mine detection, texture analysis, 3-dimensional objects recognition. Mathematical modelling in Neural network has been based on neurons that are different from real biological neurons and from the functioning of simple electronic circuit. The neuron model is made up of four basic components- an input vector, a set of synaptic weights, summing function with an activation and an output. The input of each neuron comes from two sources- external input I_i and inputs from other neurons. The total input to neuron i is the input to i = $H_i = I_i + T_{ij}V_j$ where I_i are external input and T_{ij} are synaptic interconnection strength from neuron j to neuron i. This paper discusses applications of neural networks in modeling and finding solution of various differential equations.

2. SOLUTION FOR SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS

An ODE, a system of ODE and PDE's with initial and boundary value problems are solved using artificial Neural networks. A trial solution of the differential equation is comprised of two parts. The first part is so obtained as not to affect the initial or boundary conditions, while the other part contains feed forward Neural networks containing adjustable weights (Haydar Akca M. S.), (Ravi Agarwal, 2018).Consider a system of K first order ODEs in general subject to certain initial conditions is shown in equation (1).

$$\frac{d\varphi_i}{dt} = f_i(x, \varphi_1, \varphi_2 \ldots \ldots .. \varphi_k) \quad (1)$$

With $\varphi_i(0) = A_i$ $(i = 1,2, \ldots \ldots .. k)$, we consider one neural network for each trial solution φ_{t_i} which is written in equation (2)

$$\varphi_{t_i}(x) = A_i + x\, N_i(x\, , p_i) \quad (2)$$

Where $N_i(x\, , p_i)$ is the output of a feedforward neural network with one input unit for x and weights p_i S.The error quantity to be minimized is given in equation (3)

$$[p] = \sum_{k=i}^{K} \sum_i \left\{\frac{d\varphi_{t_k}}{dx} - f_k\left(x_i, \varphi_{t_1} \ldots \ldots \ldots \varphi_{t_k}\right)\right\}^2 \quad (3)$$

Since $\frac{d\varphi_{t_i}(x)}{dx} = N_i(x, p_i) + x\frac{d}{dx}N_i(x, p_i)$, it is straight forward to compute the gradient of the error with respect to the parameters p_i

3. NEURAL NETWORK FOR SOLVING PARTIAL DIFFERENTIAL EQUATION

The neural network is trained in an unsupervised manner using error function that is derived from the differential equation itself and the boundary conditions. Figure 1 represents the schematic diagram of Neural Network. The neural network solutions are more accurate as compared to solution obtained with numerical methods. (Karami, 2007), (Haydar Akca M. S.) To use an unsupervised feed forward neural network to solve Burger's equation this is the one-dimensional quasilinear parabolic partial differential equation. Consider the equation of the form

$$u_t + uu_x = vu_{xx} \quad a < x < b, \quad t > 0 \tag{4}$$

$$u(x, 0) = g(x) \ a < x < b \tag{5}$$

$u(a, t) = g_1(t)$ and $u(b, t) = g_2(t)$,where $v > 0$ is the coefficient of the Kinematics Viscosity of the fluid. This equation intended as an approach to study turbulence, shock waves and the Gas dynamics. Let$F(x, t) = vu_{xx} - u_t - uu_x = 0$. This feed forward neural consists of two inputs x_1 and x_2 ,H hidden layers and one output u. The sigmoid function is used for hidden layers and the linear function is used for output neuron. The energy function is assumed as

$$E = \sum_{i=1}^{4} E_r \tag{6}$$

Where $E_1 = |F(x, t)|^2$, $E_2 = |u(x, 0) - G(x)|^2$, $E_3 = |u(a, t) - g_1(x)|^2$ and $E_4 = |u(b, t) - g_2(x)|^2$. We get an accurate solution, if the energy function or error function reduces close to 0. This is possible iff each term in the right hand side should be identically equal to 0. If $E_1 = 0$, it ensures that $u(x, t)$ satisfies the equation whereas reducing E_2, E_3 and E_4 to 0 implies that $u(x, t)$ is unique.

The given PDE is reformulated as follows

$$u = \sum_{i=1}^{H} f^1 (x_1 w^1(i,1) + x_2 w^1(i,2) + b^1) w^2(1,i) + b^2 \tag{7}$$

$$\frac{\partial u}{\partial x_1} = \sum_{i=1}^{H} f^{1'} (x_1 w^1(i,1) + x_2 w^1(i,2) + b^1) w^1(i,1) w^2(1,i) \tag{8}$$

$$\frac{\partial u}{\partial x_2} = \sum_{i=1}^{H} f^{1'} (x_1 w^1(i,1) + x_2 w^2(i,1) + b^1) w^1(i,2) w^2(1,i) \tag{9}$$

$$\frac{\partial^2}{\partial x^2} u = \sum_{i=1}^{H} f^{1''} (x_1 w^1(i,1) + x_2 w^1(i,2) + b^1) w^1(i,1) w^1(i,1) w^2(1,i) \tag{10}$$

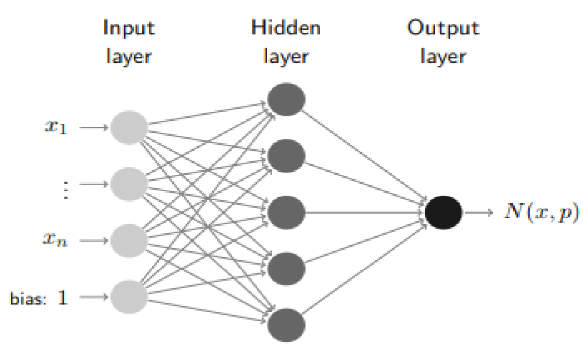

Figure 1. Schematic Diagram of Neural Network with (n+1) input nodes with 'h' hidden nodes and 1 output node(N)

Where f^1 is the sigmoid function and $W(i,j)$ is the weight between j[th] neuron of a layer and i[th] neuron of next layer.The trained network creates solutions including points those are not considered during state of time. The accuracy can be efficiently controlled by changing the number of hidden layer neurons.

4. HOPFIELD'S NEURAL NETWORK MODELLED BY CAPUTO FRACTIONAL DIFFERENTIAL EQUATIONS

Among the most popular models in the literature of Artificial Neural network is Hopfield neural network. This model is described by a set of differential equations with delays, namely, Functional differential Equations. The attractive problems in the dynamic behavior of Hopfield neural network are those of existence, uniqueness and global asymptotic stability of the equilibrium point.

1. Consider the general Hopfield's neural networks with impulses and bounded delays and distributed delays (Ravi Agarwal, 2018).

$$ {}_{t_k}^{C}D_t^q x_i(t) = c_i(t)x_i(t) + \sum_{j=1}^{n} a_{ij}(t)\, f_j\big(x_j(t)\big) + \sum_{j=1}^{n} b_{ij}(t)\, g_j\left(x_j\left(t-\tau_j(t)\right)\right) + \sum_{j=1}^{n} d_{ij}(t)\int_{-r}^{+r} K_{ij}(s)h_j\left(x_j(t+s)\right)ds\, I_i(t) \quad (11) $$

For $\in (t_k,t_{k+1}]$, k=0,1,2…………,$([\Delta x_i(t)])_{t=t_k} = I_{k,j}\big(x_i(t_k-0)\big)$, k=1,2,3……………$x(t) = \varphi_i^0(t-t_0)$ I =1,2,…….n. Where n represents the number neutrons in the network, $x_i(t)$ represents the pseudostate state variable ,q$\in (0,1)$, $c_i(t) > 0$ is the self regulating parameter of i-th neuron , $a_{ij}(t), b_i(t)$ correspond to the synaptic connection strength of the i-th neuron to the jth neuron at the time t and $t-\tau_j(t). f_j(x), g_j(x), and\ h_j(x)$ are activation functions, I_i is an external bias vector , $\tau_j(t)$ is the transmission delay along the axis of the j-th unit and satisfies $0< \tau_j(t) \le r,\ the\ functions\ \varphi_{,ki}$ are the impulsive functions giving the impulsive perturbation of the ith neuron on the point t_k. $x_i(t_k-0)$ and $x_i(t_k+0)$ are the state of the i-th neuron before and after the impulsive perturbation at time t_k. $K_{ij}\ (s)$ is the delay kernal with $\int_{-r}^{+0} K_{ij}(s)\,ds = 0, \varphi_i^0 \epsilon C([-r,0],R) \quad i = 1,2 \ldots \ldots n$

2. Consider the system of integro-differential equations as a model of Hopfield neural network with continuously distributed delays over a certain period of time (Haydar Akca R. A., 2004), (Kang, 1990).

$$ \dot{x} = -a_i x_i(t) + \sum_{j=1}^{n} a_{ij}(t)\, g_j + \sum_{j=1}^{n} b_{ij}(t)\int_{-\infty}^{t} K_{ij}(t-s)g_j\left(x_j(s)\right) + I_i(t) \qquad t \ge 0 \quad (12) $$

$$ x_i(t) = \varphi_i(t), \qquad -\infty < t \le 0\ , \quad i = 1,2 \ldots\ldots\ldots.n \quad (13) $$

The impulsive conditions are $\Delta x_i(t) = I_i\big(x_i(t)\big),\ t = t_k, k = 1,2 \ldots$..The nonlinear neural activation function $f_i\ i \in Z^{+}$, are chosen to be continuous and differentiable. It satisfies the following conditiona, $f_i(x)$ is bounded above by +1 and bounded below by -1, $f_i(x) = 0$ at a unique point $x = 0$, $f_i'(x)$ has a global maximum value of 1 at a unique point $x = 0$.Here, $\Delta x_i(t)$ are the impulses at the moment t_k and t_k is a strictly increasing sequence such it tends to ∞ as $k \to \infty$, $x_i(t)$ corresponds to the membrane potential of the unit i to the unit j, b_{ij} denotes the synaptic connection weight of the unit j to the unit I, I_i corresponds to the external bias, the coefficient a_i is the rate with which unit self regulates or resets its potential when isolate from other units and inputs, φ_i is continuous for t, K_{ij} is the delay kernal and are bounded and continuous with $\int_0^\infty K_{ij}(s)\,ds = 1$.

5. ESTMATING THE SOLUTION OF FUZZY DIFFERENTIAL EQUATION USING BERNSTEIN NEURAL NETWORKS.

The uncertain nonlinear systems can be modelled with fuzzy equations or fuzzy differential equation by incorporating the fuzzy set theory. The solutions of them are applied to analyze many engineering problems. The solutions of FDEs are approximated by Bernstein neural network. Initially the FDE is transformed into four ordinary differential equations. Then neural networks are constructed with the structure of ODEs. With the back-propagation method for Z-number variables, the neural networks are trained. The theory of analysis and simulation results show that Bernstein neural networks are effective in approximating the solution of FDEs based on the Z-numbers (Raheleh Jafari, 2017).Consider the uncertain nonlinear system of FDE

$$\frac{dx}{dt} = f(t,x), \qquad x \in \mathbb{R}^n \tag{14}$$

Where $x \in \mathbb{R}^n$ is the Z-number variable, $f(t,x)$ is a Z-number vector function, $\frac{dx}{dt}$ is the derivative associated to the Z-number variable.The Bernstein neural network uses the following Bernstein polynomial,

$$B(x_1,x_2) = \sum_{i=0}^{N} \sum_{j=0}^{M} \binom{N}{i}\binom{M}{j} W_{ij} x_{1i} (T - x_{1i})^{N-i} x_{2j} \left(1 - x_{2j}\right)^{M-j} \tag{15}$$

Where W_{ij} is the Z-number coefficient.This polynomial is considered as a neural network consists of two inputs x_{1i} *and* x_{2j} and one output y.The four ODEs equivalent to the given fuzzy differential equation are the following

$$\frac{dx}{dt} = \underline{f}\left[t, \underline{x}(\zeta,\alpha), \overline{x}(\zeta,\alpha)\right] \tag{16}$$

$$\frac{d\overline{x}}{dt} = \underline{f}\left[t, \underline{x}(\zeta,\alpha), \overline{x}(\zeta,\alpha)\right] \tag{17}$$

$$\frac{d\overline{x}}{dt} = \overline{f}\left[t, \overline{x}(\zeta,\alpha), \underline{x}(\zeta,\alpha)\right] \tag{18}$$

$$\frac{dx}{dt} = \overline{f}\left[t, \overline{x}(\zeta,\alpha), \underline{x}(\zeta,\alpha)\right] \tag{19}$$

The Bernstein neural network is used to approximate the solutions of four ODEs given above.

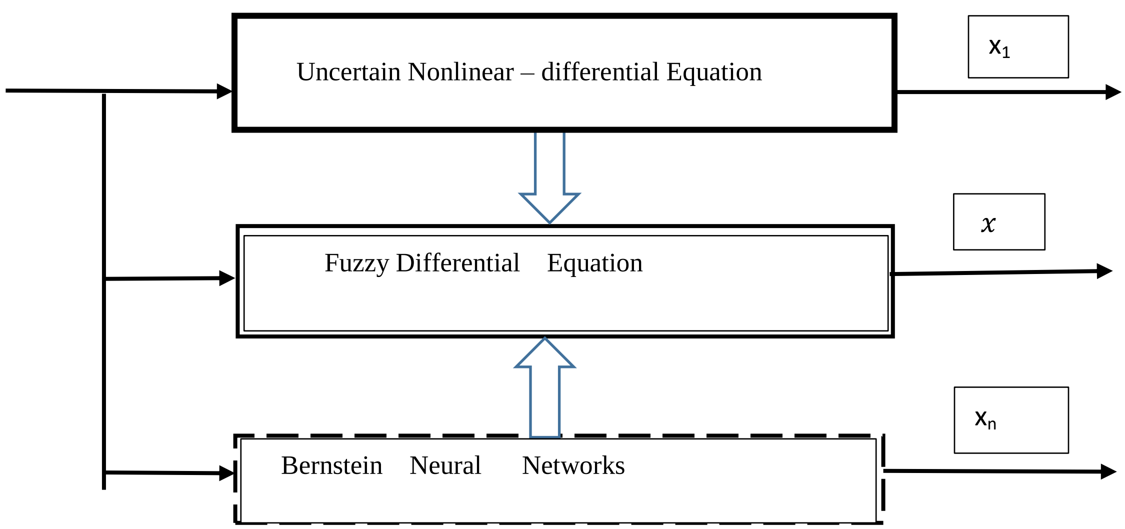

Figure 2. Nonlinear model with Fuzzy Differential Equation.

If x_1 and x_2 in the Bernstein polynomial are defined as, x_1 is time interval t and x_2 is the α -level , then the solution of FDE in the form of the Bernstein neural network is

$$x_m(t,\alpha) = x_m(0,\alpha) + \sum_{i=0}^{N} \sum_{j=0}^{M} \binom{N}{i}\binom{M}{j} W_{ij}\, t_i (T - t_i)^{N-i} \alpha_j \left(1 - \alpha_j\right)^{M-j} \tag{20}$$

Where $x_m(0,\alpha)$ is the initial condition of the solution based on the Z-numer.

EXAMPLE

5.1. Problem on Ordinary Differential Equation

In this model problem, we have multilayered perceptron having one hidden layer with 10 hidden units and one linear output unit (Haydar Akca M. S.). The sigmoid activation of each hidden unit is $\sigma(x) = \frac{1}{1+e^{-x}}$. Consider the following equation

$$\frac{d^2}{dx^2}\varphi + \frac{1}{5}\frac{d}{dx}\varphi + \varphi = -\frac{1}{5}e^{-\frac{x}{5}}\cos x \tag{21}$$

The boundary conditions are $\varphi(0) = 0 \; and \; \varphi(1) = \sin(1)\, e^{-\frac{x}{5}}$ with $x \in [0,1]$.The exact solution of this equation is $\varphi(x) = e^{-\frac{x}{5}}\sin x$ and the trial solution is of the form

$$\varphi_t(x) = x\sin(1)\, e^{-\frac{x}{5}} + x(1-x)N(x,p) \tag{22}$$

We used a grid of 10 equidistant points and the plots of the deviation from the exact solution for the boundary value problem.

5.2. Problem on Partial Differential Equation

Consider the Elliptic Laplace's equation: $\nabla^2\varphi(x) = 0, \forall\, x\epsilon D$. The boundary conditions are chosen as $\varphi(x) = 0$, $x\epsilon\{(x_1,x_2)\epsilon D/x_1 = 0, x_1 = 1, x_2 = 0\}$ $\varphi(x) = \sin\pi x_{1,}\forall$ $x\epsilon\{(x_1,x_2)\epsilon\partial D/, x_2 = 1\}$. the analytic solution is

$$\varphi(x) = \frac{1}{e^{\pi}-e^{-\pi}}\sin\pi x_1 \left(e^{\pi x_2} - e^{-\pi x_2}\right) \tag{23}$$

By using the BC's , the trial solution was constructed as

$$\varphi_t(x,v,W) = x_2 \sin\pi x_1 + x_1\left(x_1 - 1\right) + x_2\left(x_2 - 1\right)N(x,v,W) \tag{24}$$

When K=16 and H=6 ,the numerical solution and the corresponding analytic solution are in good agreement ,obtaining maximum error of about 2.10^{-4} (Kiene)

5.3. Matlab ToolBox for Solving Differential Equations

The table 1 shows matlab code for solving differential equations using Symbolic Math Toolbox™. The first column represents the sample differential equations and their corresponding matlab code given second column.

Differential Equation	**MATLAB® Commands**
$\frac{dy}{dt} + 4y(t) = e^{-t}$ $y(0) = 1$	syms y(t) DE = diff(y)+4*y == exp(-t); cond = y(0) == 1; ysolution(t) = dsolve(DE,cond), ysolution(t) =exp(-t)/3 + (2*exp(-4*t))/3
$2x^2\frac{d^2y}{dx^2} + 3x\frac{dy}{dx} - y = 0$	syms y(x) DE = 2*x^2*diff(y,x,2)+3*x*diff(y,x)-y == 0; ysolution (x) = dsolve(DE)ysolution (x) =C2/(3*x) + C3*x^(1/2)
$\frac{d^2y}{dx^2} = xy(x)$	syms y(x) DE = diff(y,x,2) == x*y; ysolution(x) = dsolve(DE)ysolution(x) =C1*airy(0,x) + C2*airy(2,x)

Traditional methods such as finite elements, finite volume, and finite differences solve the differential equations over this discretization weakly. Generally, these methods are adequate and effective in many scientific and engineering applications. The one limitation of using this is that the obtained solutions are discrete or sometimes it have limited differentiability. To address this issue, when numerically solving differential equations, one can implement a different method which relies on neural networks. The purpose of this study is to outline the MATLAB oriented method for solving it.

6. CONCLUSION

In this paper, general features of neural algorithm for solving differential equations are discussed. The artificial neural network solutions are more accurate as compared to the solutions obtained with numerical methods. It helps us achieve solutions faster without wasting memory space and computational time. A comparison to the exact solution reveals that it reduces the complexity of the problems due to the parallel structure of the network.

REFERENCES

Haydar Akca, M. S. Numerical methods for solution of impulsive differential equations and stability analysis. 99(12), 1955-1970

Haydar Akca, R. A. (2004). Neural Networks: Modelling with Impulsive Differential Equations. Antalaya, Turkey – Dynamical Systems and Applications, (pp. 32-47).

I.E. Lagaris, A. L. (1997). Artificial Neural Networks for solving Ordinary and Partial differential equations.

Kang, H. L. (1990). Neural Algorithm for solving Differential Equations. Journal of Computational Physics, 110-131.

Karami, M. H. (2007). Feedforward Neural network for solving partial differential equations. Journal of Applied Sciences, 7(19), 2812-2817.

Kiene, M. C. Solving differential equations using neural networks. Machine Learning Project.

Raheleh Jafari, W. Y. (2017). Numerical Solution of Fuzzy Differential Equations with Z-numbers using Bernstein Neural networks. International Journal of Computational Intelligence Systems, 10, 1226-1237.

Ravi Agarwal, S. H. (2018). Global Mittag-Leffler Synchronization for Neural Networks modelled by Caputo Fractional Differential Equations with Distributed delays.