Functionally Weighted Neural Networks with Infinite Number of Neurons

I. E. Lagaris

CSE, University of Ioannina

Abstract

A new type of Neural Network is presented, with a single hidden layer and an infinite number of neurons. To render the transition to the continuum, a neuron density is introduced, the network weights become functions of a continuous variable, and the conventional sum is replaced by an integral.

Talk Structure

Feed Forward Neural Networks

There is a plethora of Feed Forward Neural Networks that differ in:

Architecture: *Shallow*
Deep

Number of Nodes: *Few or Many*

Activation: *Sigmoid:* $\sigma(x) = [1 + exp(-x)]^{-1}$
$\tanh(x) = 2\sigma(2x) - 1$
Gaussian: $G(x, \mu, \sigma) = e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$
Multiquadric: $\sqrt{1 + x^2}$
Thin plate spline: $x^2 \ln(x)$
Legendre, Chebychev, Bernstein, ...

Why a new Neural Network, and what is expected from it?

A new network in order to be competitive should offer:

- Higher Accuracy
- Parametric Economy
- Enhanced Interpolation Generalization
- Enhanced Extrapolation Generalization

"Parametric Economy" and "Generalization" are intimately correlated !!!

The Way to Higher Accuracy

It is proved[1] that single hidden layer networks can approximate any function, to **any desired degree of accuracy** provided that **sufficient number of neurons** are available.

Hence, to obtain ultimate accuracy,
the number of neurons should tend to Infinity.

$$\infty$$

[1] *K. Hornik, M. B. Stinchcombe, H. White, Neural Networks 2(1989)359-366, Multilayer feedforward networks are universal approximators*

Gaussian RBF Networks

A Gaussian RBF Network with K nodes (neurons), is given by:

$$N_G(x,\theta) = \sum_{i=1}^{K} A_i e^{-\frac{1}{2}\left(\frac{|x-\mu_i|}{\sigma_i}\right)^2} \equiv \sum_{i=1}^{K} A_i G(x,\mu_i,\sigma_i)$$

where $\boldsymbol{\theta}$ stands collectively for all $\{A_i, \mu_i, \sigma_i\}$.

What happens when $K \to \infty$?

Disaster ... at First Sight

1. Number of parameters (weights): **Infinite !!!**
2. Computational Task: **Impossible !!!**
3. Approximation: **Exact but Worthless !!!**
4. Generalization: **Infeasible !!!**

With four parameters I can fit an Elephant,
and with five I can make him wiggle his Trunk.

John von Neumann

Transition to the Continuum[2]

In Physics this is a familiar limiting procedure ...

- The continuum limit of a chain, is a string.
- Discrete points are replaced by a point density.
- Differences become Derivatives.
- Indexed quantities become functions.
- Sums become Integrals.

[2] K. Blekas and I. E. Lagaris, *"Artificial neural networks with an infinite number of nodes"*. IOP Conf. Series: Journal of Physics: Conf. Series 915 (2017) 012006

Functionally Weighted Networks

Standard RBF: $N_G(x,\theta) = \sum_{i=1}^{K} A_i e^{-\frac{1}{2}\left(\frac{|x-\mu_i|}{\sigma_i}\right)^2}$

Introduce the neural node density: $\rho(s) \geq 0, \; s \in S \subset R$

Such that: $K = \int_S \rho(s) ds \to \infty$

- $A_i \to A(s)$ • $\mu_i \to \mu(s)$ • $\sigma_i \to \sigma(s)$ • $\sum_i \to \int_S ds\rho(s)$

FW-RBF:

- $N_G(x,\theta) \to N_{FW}(x,\theta) \equiv \int_S ds\rho(s)A(s)e^{-\frac{1}{2}\left(\frac{|x-\mu(s)|}{\sigma(s)}\right)^2}$

Choices for S and $\rho(s)$

Multitude of choices that satisfy: $\int_S ds\rho(s) \rightarrow \infty,\ \rho(s) \geq 0$

1. $S = (-\infty, \infty),\ \rho(s) = 1$
2. $S = [0, 1],\ \rho(s) = s^{-1}$
3. $S = [-1, 1],\ \rho(s) = (1 - s^2)^{-1}$
4. $\dots \qquad \dots \qquad \dots$

We have considered the third option: $\rho(s) = \dfrac{1}{1 - s^2}$, with $s \in [-1, 1]$

$$\boldsymbol{N_{FW}(x, \theta) \equiv \int_{-1}^{+1} \frac{ds}{1 - s^2} A(s) e^{-\frac{1}{2}\left(\frac{|x - \mu(s)|}{\sigma(s)}\right)^2}}$$

Gauss-Chebyshev Quadrature

A Technical Note

The Gauss-Chebyshev rule, known to be highly accurate, is given by:

$$\int_{-1}^{+1} \frac{f(s)}{\sqrt{1-s^2}} ds \approx \frac{\pi}{N} \sum_{i=1}^{N} f(s_i)$$

where: $s_i = \cos\left(\frac{2i-1}{2N}\pi\right), \ \forall\ i = 1, 2, \cdots, N$

FW-RBF Final Form

Setting: $w(s) \equiv \frac{A(s)}{\sqrt{1-s^2}}$, the expression for the FW-RBF becomes:

$$\boldsymbol{N_{FW}(x,\theta) = \int_{-1}^{+1} \frac{ds}{\sqrt{1-s^2}} w(s) e^{-\frac{1}{2}\left(\frac{|x-\mu(s)|}{\sigma(s)}\right)^2}}$$

Remaining task is to choose the functions $\boldsymbol{w(s), \mu(s), \sigma(s)}$.

Parametrize-Economize

Let the data dimension be d. Then $\mu = (\mu_1, \cdots, \mu_d)^T \in R^d$.

Polynomial forms:

- $w(s) = \sum_{i=0}^{L_w} w_i s^i$
- $\mu_m(s) = \sum_{i=0}^{L_\mu} \mu_{mi} s^i$
- $\sigma(s) = \sum_{i=0}^{L_\sigma} \sigma_i s^i$

Total number of parameters: $\boldsymbol{L = L_w + d \times L_\mu + L_\sigma + d + 2}$

Ellipsoidal forms:

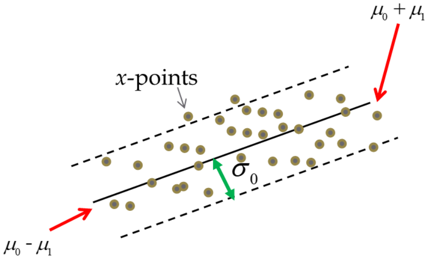
$\mu_i(s) = u_i + v_i \dfrac{s + b_i}{\sqrt{\sum_{k=1}^{d}(s+b_k)^2}}$, $w(s)$ and $\sigma(s)$ as above.

Total number of parameters: $\boldsymbol{L = L_w + L_\sigma + 3 \times d + 2}$

The number of adjustable parameters is certainly finite !!!

Simple Cases

For $L_\mu = 1$ and $L_\sigma = 0$,
$\mu(s) = \mu_0 + s\mu_1$ and $\sigma(s) = \sigma_0$



The locus of $\mu(s)$, the width σ_0, and the data points.

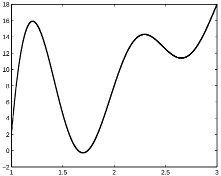
Posteriori Ascertainments

- Performed tests using data sets created by known functions.
- Each set was split in two subsets for **Training** and **Testing**.
- The training was performed both with and without “noise”.
- The testing subset remained clean (noise free).

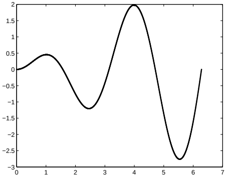
Our Findings:

- Generalization in interpolating is superior.
- The generalization performance relative to other networks, increases with the noise level. (*Noise Filter*).
- FWNN is by far more economical compared to other networks.
- The generalization in extrapolating, clearly has an edge.

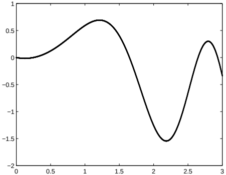
Test functions in 1-d



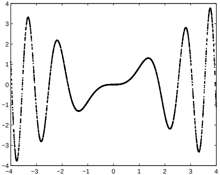
(a) $f(x) = 2x^2 + \exp(\pi/x)\sin(2\pi x)$



(b) $f(x) = x\sin(x)\cos(x)$



(c) $f(x) = \sin(x^2) - 0.25x$



(d) $f(x) = x\sin(x^2)$

Method	NMSE over the TEST set	
	medium noise	**high noise**
dataset 1(a)		
FWNN	0.63	**1.43**
MLP (best)	**0.59** ($K = 30$)	1.73 ($K = 30$)
RBF (best)	1.17 ($K = 10$)	1.78 ($K = 10$)
dataset 1(b)		
FWNN	**0.04**	**0.12**
MLP (best)	2.92 ($K = 100$)	5.43 ($K = 100$)
RBF (best)	1.19 ($K = 10$)	3.05 ($K = 10$)

FWNN configuration: $L_w = 5,\ L_\mu = 1,\ L_\sigma = 1.$

Number of Parameters. FWNN: **10**, MLP: 90/300, RBF: 30

NMSE stands for the "Normalized Mean Squared Error":

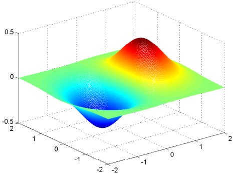
$$E_{NMSE}(\theta) = \frac{1}{M}\sum_{i=1}^{M}\left(\frac{N(x_i,\theta) - f(x_i)}{\max(1, |f(x_i)|)}\right)^2 \times 100$$

Method	NMSE over the TEST set	
	medium noise	**high noise**
dataset 1(c)		
FWNN	**0.03**	**0.24**
MLP (best)	3.67 ($K = 30$)	5.71 ($K = 10$)
RBF (best)	3.83 ($K = 20$)	6.55 ($K = 50$)
dataset 1(d)		
FWNN	**1.29**	**2.01**
MLP (best)	23.96 ($K = 100$)	48.19 ($K = 100$)
RBF (best)	3.47 ($K = 80$)	5.77 ($K = 80$)

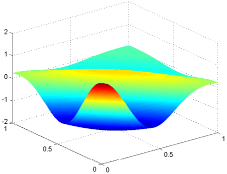
FWNN configuration: $L_w = 5,\ L_\mu = 1,\ L_\sigma = 1.$

Number of Parameters. FWNN: **10**, MLP: 90/300, RBF: 60/240

Test Functions in 2-d: Exponential and Gabor functions

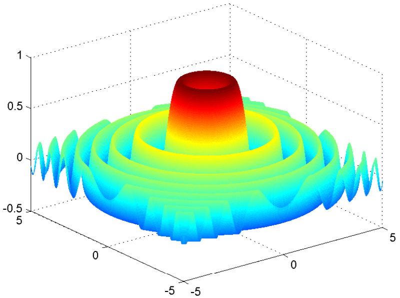


(a) $f(x_1, x_2) = x_1 \exp(-(x_1^2 + x_2^2))$



(b) $f(x_1, x_2) = \frac{\pi}{2} \exp(-2(x_1^2 + x_2^2)) \cos(2\pi(x_1 + x_2))$

The Mexican Hat Function



$$\text{(c)}\ f(x_1, x_2) = \frac{\sin(x_1^2 + x_2^2)}{\sqrt{x_1^2 + x_2^2}}$$

In each case 100 training and 1000 testing points were used.

Method	NMSE over the TEST set	
	medium noise	**high noise**
	dataset 2(a)	
FWNN	**11.14**	**22.83**
MLP (best)	19.84 ($K = 10$)	71.84 ($K = 10$)
RBF (best)	11.98 ($K = 50$)	51.73 ($K = 50$)
	dataset 2(b)	
FWNN	**1.55**	**4.66**
MLP (best)	2.34 ($K = 100$)	7.95 ($K = 100$)
RBF (best)	1.69 ($K = 50$)	8.11 ($K = 30$)
	dataset 2(c)	
FWNN	**68.99**	**69.82**
MLP (best)	84.97 ($K = 100$)	110.71 ($K = 100$)
RBF (best)	80.42 ($K = 80$)	86.18 ($K = 80$)

Number of Parameters. FWNN: **12**, MLP: 40/400, RBF: 120/200/320

Extrapolation Performance

Extrapolation is connected to prediction.
Prediction is important !!!

Make me a **prophet**,
and I will make you **rich !!!**

- Does the FWNN extrapolate well ?
- Is there a fair systematic comparison procedure ?
- How does FWNN compare to the "competition" ?

Comparison Setting

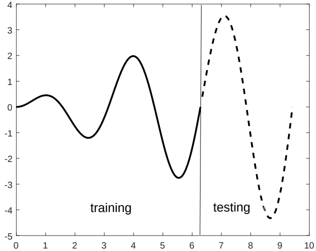
- Pick a test function $f(x)$.
- Choose 150 successive equidistant points.
- Train the networks (FWNN, MLP, RBF) using the first 100 points.
- Use the last 50 points: $x_1, \cdots, x_{50}$, for testing the extrapolation.

Extrapolation measure: $r_i \equiv \dfrac{|f(x_i) - N(x_i, \theta)|}{\max(1, |f(x_i)|)}$, the relative deviation.

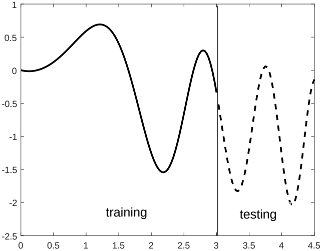
For satisfactory extrapolation, r_i should be small.
Let $d \in (0, 0.25]$ be an acceptable upper bound for r_i, i.e. $r_i \leq d$.
Determine J such that: $r_i < d$, $\forall i \in [1, J]$ and $r_{J+1} \geq d$.

The network with the highest index *J*,
is the extrapolation Winner

Extrapolation Test Functions

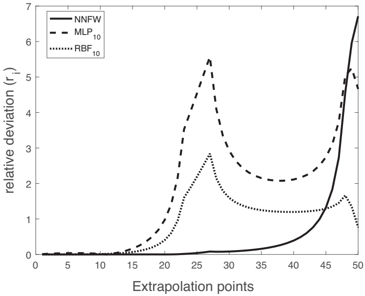


$$f_1(x) = x\sin(x)\cos(x)$$

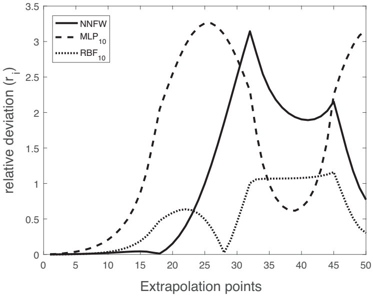


$$f_2(x) = sin(x^2) - 0.25x$$

Extrapolation Performances



for: $f_1(x) = x\sin(x)\cos(x)$



for: $f_2(x) = sin(x^2) - 0.25x$

Extrapolation Comparison

Extrapolation index *J*

Network Architecture	**Deviation bound:** d				
	0.05	0.10	0.15	0.20	0.25
$f_1(x) = x\sin(x)\cos(x)$					
FWNN	$\mathbf{24} \pm 3$	$\mathbf{25} \pm 3$	$\mathbf{35} \pm 2$	$\mathbf{37} \pm 2$	$\mathbf{38} \pm 2$
MLP 10 nodes	12 ± 2	14 ± 2	15 ± 3	16 ± 1	16 ± 1
RBF 10 nodes	14 ± 2	16 ± 2	17 ± 2	18 ± 1	19 ± 1
$f_2(x) = sin(x^2) - 0.25x$					
FWNN	$\mathbf{18} \pm 1$	$\mathbf{20} \pm 1$	$\mathbf{21} \pm 1$	$\mathbf{21} \pm 1$	$\mathbf{21} \pm 1$
MLP 10 nodes	6 ± 2	8 ± 1	9 ± 1	10 ± 1	11 ± 1
RBF 10 nodes	11 ± 1	13 ± 2	14 ± 2	15 ± 1	16 ± 1

Training Techniques

"Training" a Neural Network, is an optimization problem with the following "Sum-Of-Squares" objective function:

$$E(\theta) = \sum_{i=1}^{M} [N(x_i, \theta) - y_i]^2 \equiv \sum_{i=1}^{M} [R(x_i, \theta)]^2$$

Its gradient and Hessian given by:

$$\nabla_\theta E(\theta) = 2 \sum_{i=1}^{M} R(x_i, \theta) \nabla_\theta R(x_i, \theta)$$

$$\nabla_\theta^2 E(\theta) = 2 \sum_{i=1}^{M} [\nabla_\theta R(x_i, \theta) \nabla_\theta R(x_i, \theta)^T + R(x_i, \theta) \nabla_\theta^2 R(x_i, \theta)]$$

Small Residual Problems

If the model, i.e. the Network $N(x, \theta)$, is proper, then near the minimum point $\boldsymbol{\theta^*}$, i.e. for $||\boldsymbol{\theta} - \boldsymbol{\theta^*}|| \leq \boldsymbol{\epsilon}$, $\boldsymbol{R(x_i, \theta) \approx 0}$.

This is called a "**Small Residual Problem**", and in this case the Hessian may be approximated using **first derivatives only** as:

$$\nabla^2_\theta E(\theta) \approx 2 \sum_{i=1}^{M} \nabla_\theta R(x_i, \theta) \nabla_\theta R(x_i, \theta)^T$$

The indicated optimization methods therefore, belong to the so called "Gauss-Newton" class, either within the "Trust-Region" framework (Levenberg-Marquardt), or with the "line-search" approach.

Large Residual Problems

When for $||\boldsymbol{\theta} - \boldsymbol{\theta}^*|| \leq \epsilon, \; \boldsymbol{R}(\boldsymbol{x_i}, \boldsymbol{\theta}) \gg \boldsymbol{0}$, as for example in the case of "very" noisy data, we have a "**Large Residual Problem**". In this case the "Gauss-Newton" approximation is not valid.

Appropriate methods are:

- "Modified Newton"
- "Quasi-Newton" (SR1, BFGS)
- "Limited Memory Quasi-Newton"
- "Conjugate Gradient" (Polak-Ribiere, Dixon, ...)
- "Hybrid Methods" (Fletcher & Xu[3])

[3] *Hybrid Methods for Nonlinear Least Squares*. IMA Journal of Numerical Analysis, 7 (1987) 371–389

Solving ODEs & PDEs with FWNN

ANNs have been used in the past to solve ODEs and PDEs.
A set of problems was considered and solved in a work[4] entitled:

"*Artificial Neural Networks for solving ordinary and partial differential equations*".

These problems have since been used as benchmarks by several authors who were developing methods for ODEs and/or PDEs, using various kinds and architectures of neural networks.

We have applied the same methodology using the FW-RBF Network instead of the MLP, on two of these problems (Problems #4 and #5).

[4] *IEEE Transactions on Neural Networks, 9 (1998) 987-1000*

System of ODEs. Problem #4

$$\frac{d\Psi_1(x)}{dx} = \cos(x) + \Psi_1^2(x) + \Psi_2(x) - (1 + x^2 + \sin^2(x))$$
$$\frac{d\Psi_2(x)}{dx} = 2x - (1 + x^2)\sin(x) + \Psi_1(x)\Psi_2(x)$$
$$\Psi_1(0) = 0, \quad \Psi_2(0) = 1, \quad x \in [0, 3]$$

Exact solution: $\Psi_1(x) = \sin(x), \Psi_2(x) = 1 + x^2$

Trial Solution: $\Psi_{1t}(x) = xN_1(x, \theta_1), \; \Psi_{2t}(x) = 1 + xN_2(x, \theta_2)$

Number of Points: 10 for Training and 100 for Testing.

Preliminary Results

Network	# of Parameters	MAD1	MAD2
MLP	30 $(= 10 \times 3)$	2.0E-5	8.0E-5
FWNN	**8** $(= 3 + 3 + 2)$	5.0E-5	7.0E-5

PDE in 2-d. Problem #5

$$\nabla^2\Psi(x,y) = e^{-x}(x-2+y^3+6y), \quad (x,y) \in [0,1]\otimes[0,1]$$

Dirichlet BCs: $\Psi(0,y) = y^3, \quad \Psi(1,y) = \dfrac{1+y^3}{e}$

$$\Psi(x,0) = xe^{-x}, \Psi(x,1) = (x+1)e^{-x}$$

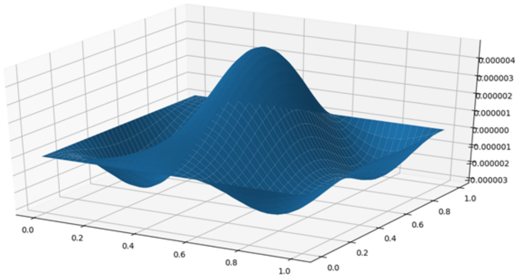
with exact solution: $\Psi(x,y) = e^{-x}(x+y^3)$.

Trial Solution: $\Psi_t(x,y) = A(x,y) + x(1-x)y(1-y)N(x,y,\theta)$

$$\boldsymbol{A(x,y) = (1-x)y^3 + x\frac{1+y^3}{e} + (1-y)x\left(e^{-x} - e^{-1}\right) + y\left[(x+1)e^{-x} - \left(1 - x + 2xe^{-1}\right)\right]}$$

$\Delta\Psi(x,y) = \Psi(x,y) - \Psi_t(x,y)$

Used a FWNN with 11 parameters $(3 + 2 \times 3 + 2)$



Plot of the difference between the exact and the calculated solutions.

A mesh of 100 points (10×10) was used for training, and a mesh of 900 points (30×30) for testing. Absolute mean deviation $\approx 1.3 \times 10^{-6}$.

Conclusions

The main features of the FWNN may be summarized as:

- “Economic” in the number of parameters.
- Excellent generalization while interpolating.
- Superior extrapolation capability.

Work to be done

- Only polynomials have been tried up to now for $\boldsymbol{w(s)}, \boldsymbol{\mu(s)}, \boldsymbol{\sigma(s)}$. More forms should be investigated.
- Sigmoidal FWNNs should also be explored, i.e.: $N_{FW}^{\sigma}(x,\theta) = \int_{-1}^{1} \frac{ds}{\sqrt{1-s^2}} a(s)\sigma(w^T(s)x + b(s))$
- Extend FWNN to Deep-FWNN, to explore possible benefits.