Introduction to Neural Networks: Structure and Training

Professor Q.J. Zhang

Department of Electronics
Carleton University, Ottawa, Canada

www.doe.carleton.ca/~qjz, qjz@doe.carleton.ca
A Quick Illustration Example:

Neural Network Model for Delay Estimation in a High-Speed Interconnect Network
High-Speed VLSI Interconnect Network

Driver 1
Driver 2

Receiver 1

Driver 3
Receiver 2

Receiver 3

Receiver 4
Circuit Representation of the Interconnect Network
Massive Analysis of Signal Delay
Need for a Neural Network Model

• A PCB contains large number of interconnect networks, each with different interconnect lengths, terminations, and topology, leading to need of massive analysis of interconnect networks

• During PCB design/optimization, the interconnect networks need to be adjusted in terms of interconnect lengths, receiver-pin load characteristics, etc, leading to need of repetitive analysis of interconnect networks

• This necessitates fast and accurate interconnect network models and neural network model is a good candidate
Neural Network Model for Delay Analysis
3 Layer MLP: Feedforward Computation

\[
\begin{align*}
y_j &= \sum_k W'_{jk} Z_k \\
Z_k &= \tanh(\sum_i W_{ki} x_i)
\end{align*}
\]
Neural Net Training

Training Data
by simulation/measurement
\( d = d(x) \)

Neural Network
\( y = y(x) \)

Objective:
to adjust \( W, V \) such that

\[
\text{minimize } \sum_{x} (y - d)^2
\]
Simulation Time for 20,000 Interconnect Configurations

<table>
<thead>
<tr>
<th>Method</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit Simulator (NILT)</td>
<td>34.43 hours</td>
</tr>
<tr>
<td>AWE</td>
<td>9.56 hours</td>
</tr>
<tr>
<td>Neural Network Approach</td>
<td>6.67 minutes</td>
</tr>
</tbody>
</table>
Important Features of Neural Networks

• Neural networks have the ability to model multi-dimensional nonlinear relationships

• Neural models are simple and the model computation is fast

• Neural networks can learn and generalize from available data thus making model development possible even when component formulae are unavailable

• Neural network approach is generic, i.e., the same modeling technique can be re-used for passive/active devices/circuits

• It is easier to update neural models whenever device or component technology changes
Inspiration

Mary       Lisa       John

“Stop”
“Start”
“Help”
A Biological Neuron

Figure from Reference [L.H. Tsoukalas and R.E. Uhrig, Fuzzy and Neural Approaches in Engineering, Wiley, 1997.]
Neural Network Structures
Neural Network Structures

• A neural network contains
  • neurons (processing elements)
  • connections (links between neurons)

• A neural network structure defines
  • how information is processed inside a neuron
  • how the neurons are connected

• Examples of neural network structures
  • multi-layer perceptrons (MLP)
  • radial basis function (RBF) networks
  • wavelet networks
  • recurrent neural networks
  • knowledge based neural networks

• MLP is the basic and most frequently used structure
MLP Structure

(Output) Layer L

(Hidden) Layer L-1

(Hidden) Layer 2

(Input) Layer 1
Information Processing In a Neuron

\[ \sigma(.) \]

\[ \sum \]

\[ z_i^l \]

\[ \gamma_i^l \]

\[ w_{i0}^l \]

\[ w_{i1}^l \]

\[ w_{i2}^l \]

\[ z_0^{l-1} \]

\[ z_1^{l-1} \]

\[ z_2^{l-1} \]

\[ z_{N_{l-1}}^{l-1} \]
Neuron Activation Functions

• Input layer neurons simply relay the external inputs to the neural network

• Hidden layer neurons have smooth switch-type activation functions

• Output layer neurons can have simple linear activation functions
Forms of Activation Functions: $z = \sigma(\gamma)$

Sigmoid

Arctangent

Hyperbolic tangent
Forms of Activation Functions: $z = \sigma(\gamma)$

- **Sigmoid function:**
  \[
  z = \sigma(\gamma) = \frac{1}{1 + e^{-\gamma}}
  \]

- **Arctangent function:**
  \[
  z = \sigma(\gamma) = \frac{2}{\pi} \arctan(\gamma)
  \]

- **Hyperbolic Tangent function:**
  \[
  z = \sigma(\gamma) = \frac{e^{+\gamma} - e^{-\gamma}}{e^{+\gamma} + e^{-\gamma}}
  \]
Multilayer Perceptrons (MLP):

Structure:

- **(Input) layer 1**
- **(Hidden) layer 2**
- **(Hidden) layer l**
- **(Output) layer L**

\[ x_1, x_2, x_3, \ldots, x_n \]
where: \( L = \) Total No. of the layers
\( N_l = \) No. of neurons in the layer \#\( l \)
\( l = 1, 2, 3, \ldots, L \)
\( w_{ij}^l = \) Link (weight) between the neuron \#\( i \) in the layer \#\( l \) and the neuron \#\( j \) in the layer \#(\( l-1 \))

NN inputs = \( x_1, x_2, \ldots, x_n \) (where \( n = N_1 \))

NN outputs = \( y_1, y_2, \ldots, y_m \) (where \( m = N_L \))

Let the neuron output value be represented by \( z, \)
\( z_i^l = \) Output of neuron \#\( i \) in the layer \#\( l \)

Each neuron will have an activation function:
\( z = \sigma (\gamma ) \)
Neural Network Feedforward:

Problem statement:

Given: \( x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \), get \( y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \) from NN.

Solution: feed \( x \) to layer #1, and feed outputs of layer \#l-1 to \( l \).

For \( l = 1 \):
\[
z_i^l = x_i, \quad i = 1,2,\ldots,n, \quad n = N_1
\]

For \( l = 2,3,\ldots,L \):
\[
\gamma_i^l = \sum_{j=0}^{N_{l-1}} w_{ij}^l z_j^{l-1}, \quad z_i^l = \sigma(\gamma_i^l)
\]

and solution is \( y_i = z_i^L, \quad i = 1,2,\ldots,m, \quad m = N_L \)
Question: How can NN represent an arbitrary nonlinear input-output relationship?

Summary in plain words:

Given enough hidden neurons, a 3-layer-perceptron can approximate an arbitrary continuous multidimensional function to any required accuracy.
Theorem (Cybenko, 1989):
Let $\sigma(\cdot)$ be any continuous sigmoid function, the finite sums of the form:

$$y_k = \tilde{f}_k(x) = \sum_{j=1}^{N_2} w_{kj}^{(3)} \sigma \left( \sum_{i=0}^{n} w_{ji}^{(2)} x_i \right) \quad k = 1, 2, \ldots, m$$

are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\varepsilon > 0$, there is a sum, $\tilde{f}(x)$, of the above form, for which

$$\left| \tilde{f}(x) - f(x) \right| < \varepsilon \text{ for all } x \in I_n$$

where:

$I_n$ -- $n$-dimensional unit cube $[0, 1]^n$

$x$ space: $x_i \in [0, 1], i = 1, 2, \ldots, n$

$C(I_n)$: Space of continuous functions on $I_n$.

e.g., Original problem is $y = f(x)$, where $f \in C(I_n)$, the form of $\tilde{f}(x)$ is a 3-layer-perceptron NN.
Suppose this neuron is $z_i^l$. Weights $w_{ij}^l$ (or $w_{ki}^{l+1}$) affect the figure horizontally (or vertically). Values of the $w$'s are not unique, e.g., by changing signs of the 3 values in the example above.

Standard sigmoid function

$$\frac{1}{1 + e^{-x}}$$

Hidden neuron

$$\frac{1}{1 + e^{-(0.5x - 2.5)}}$$

Q.J. Zhang, Carleton University
Illustration of the Effect of Neural Network Weights
- Example with 1 input
Illustration of the Effect of Neural Network Weights
- Example with 2 inputs

$z_1$ as a function of $x_1$ and $x_2$:

$$z_1 = \frac{1}{1+\exp(-x_1)}$$
Illustration of the Effect of Neural Network Weights
- Example with 2 inputs

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>z1</th>
<th>z2</th>
<th>z3</th>
<th>z4</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
</tr>
</tbody>
</table>

Assuming arctan activation function
Illustration of the Effect of Neural Network Weights
- Example with 2 inputs

Assuming arctan activation function
Illustration of the Effect of Neural Network Bias Parameters
- Example with 2 inputs

Assuming arctan activation function
Effect of Neural Network Weights, and Inputs on Neural Network Outputs

As the values of the neural network inputs $x$ change, different neurons will respond differently, resulting in $y$ being a “rich” function of $x$. In this way, $y$ becomes a nonlinear a function of $x$.

When the connection weights and bias change, $y$ will become a different function of $x$.

$$y = f(x, w)$$
Question: How many neurons are needed?

Essence: The degree of non-linearity in original problem.

Highly nonlinear problems need more neurons, more smooth problems need fewer neurons.

Too many neurons – may lead to over learning

Too few neurons – will not represent problem well enough

Solution: \[ \begin{align*}
\text{experience} \\
\text{trial/error} \\
\text{adaptive schemes}
\end{align*} \]
Question: How many layers are needed?

- 3 or more layers are necessary and sufficient for arbitrary nonlinear approximation
- 3 or 4 layers are used more frequently
- more layers allow more effective representation of hierarchical information in the original problem
Radial Basis Function Network (RBF)

Structure:
RBF Function (multi-quadratic)

\[ \sigma(\gamma) = \frac{1}{(c^2 + \gamma^2)^\alpha}, \quad \alpha > 0 \]
\[ \sigma(\gamma) = \exp\left(-\left(\frac{\gamma}{\lambda}\right)^2\right) \]
**RBF Feedforward:**

\[ y_k = \sum_{i=0}^{N} w_{ki} z_i \quad k = 1, 2, \ldots, m \]

where \( z_i = \sigma (\gamma_i) = e^{-\gamma_i} \)

\[ \gamma_i = \sum_{j=1}^{n} \left( \frac{x_j - c_{ij}}{\lambda_{ij}} \right)^2 \quad i = 1, 2, \ldots, N, \quad \gamma_0 = 0 \]

\( c_{ij} \) is the center of radial basis function, \( \lambda_{ij} \) is the width factor.

So a set of parameters \( c_{ij} \) \((i = 1, 2, \ldots, N, j = 1, \ldots, n)\) represent centers of the RBF. Parameters \( \lambda_{ij} \) \((i = 1, 2, \ldots, N, j = 1, \ldots, n)\) represent “standard deviation” of the RBF.

**Universal Approximation Theorem (RBF) – (Krzyzak, Linder & Lugosi, 1996):**

An RBF network exists such that it will approximate an arbitrary continuous \( y = f(x) \) to any accuracy required.
Wavelet Neural Network Structure:

For hidden neuron $z_j$: 
\[ z_j = \sigma (\gamma_j) = \psi \left( \frac{x - t_j}{a_j} \right) \]

where $t_j = [t_{j1}, t_{j2}, \ldots, t_{jn}]^T$ is the translation vector, $a_j$ is the dilation factor, $\psi (\cdot)$ is a wavelet function:

\[ \psi(\gamma_j) = \sigma (\gamma_j) = (\gamma_j^2 - n)e^{-\frac{\gamma_j^2}{2}} \quad \text{where} \quad \gamma_j = \left\| \frac{x - t_j}{a_j} \right\| = \sqrt{\sum_{i=1}^{n} \left( \frac{x_i - t_{ji}}{a_j} \right)^2} \]
Wavelet Function

- $a_1 = 5$
- $a_1 = 1$
Wavelet Transform:

$R$ – space of a real variable

$R^n$-- n-dimensional space, i.e. space of vectors of $n$ real variables

A function $f : R^n \to R$ is radial, if a function $g : R \to R$, exists such that $\forall x \in R^n \quad f(x) = g(\|x\|)$

If $\psi(x)$ is radial, its Fourier Transform $\hat{\psi}(w)$ is also radial.

Let $\hat{\psi}(w) = \eta(\|w\|)$, $\psi(x)$ is a wavelet function, if

$$C_\psi = (2\pi)^n \int_0^\infty \left| \eta(\xi) \right|^2 \frac{d \xi}{\xi} < \infty$$

Wavelet transform of $f(x)$ is:

$$w(a,t) = \int_{R^n} f(x) a^{-\frac{n}{2}} \psi \left( \frac{x-t}{a} \right) dx$$

Inverse transform is:

$$f(x) = \frac{1}{C_\psi} \int_0^\infty a^{-(n+1)} \int_{R^n} w(a,t) a^{-\frac{n}{2}} \psi \left( \frac{x-t}{a} \right) dtda$$
Feedforward Neural Networks

The neural network accepts the input information sent to input neurons, and proceeds to produce the response at the output neurons. There is no feedback from neurons at layer $l$ back to neurons at layer $k$, $k \geq l$.

Examples of feedforward neural networks:

- Multilayer Perceptrons (MLP)
- Radial Basis Function Networks (RBF)
- Wavelet Networks
Recurrent Neural Network (RNN): Discrete Time Domain

The neural network output is a function of its present input, and a history of its input and output.
Dynamic Neural Networks (DNN) (continuous time domain)

The neural network directly represents the dynamic input-output relationship of the problem. The input-output signals and their time derivatives are related through a feedforward network.
Self-Organizing Maps (SOM), (Kohonen, 1984)

Clustering Problem:
Given training data $x_k$, $k = 1, 2, \ldots, P$,
Find cluster centers $c_i$, $i = 1, 2, \ldots, N$

Basic Clustering Algorithm:
For each cluster $i$, $(i=1,2,\ldots,N)$, initialize an index set $R_i=$\{empty\}, and set the center $c_i$ to an initial guess.

For $x_k$, $k = 1, 2, \ldots, P$, find the cluster that is closest to $x_k$, i.e. find $c_i$, such that,
$$\|c_i - x_k\| < \|c_j - x_k\|, \quad \forall j, \quad i \neq j$$

Let $R_i = R_i \cup \{k\}$. Then the center is adjusted:
$$c_i = \frac{1}{\text{size}(R_i)} \sum_{k \in R_i} x_k$$

and the process continues until $c_i$ does not move any more.
Self Organizing Maps (SOM)

SOM is a one or two dimensional array of neurons where the neighboring neurons in the map correspond to the neighboring cluster centers in the input data space.

Principle of Topographic Map Information: (Kohonen, 1990)

The spatial location of an output neuron in the topographic map corresponds to a particular domain or features of the input data.
Training of SOM:

For each training sample $x_k$, $k = 1, 2, \ldots, P$, find the nearest cluster $c_{ij}$, such that

$$
\|c_{ij} - x_k\| < \|c_{pq} - x_k\|, \quad \forall p, q, p \neq i, q \neq j
$$

Then update $c_{ij}$ and neighboring centers:

$$
c_{pq} = c_{pq} + \alpha(t)(x_k - c_{pq})
$$

where

$$
|p - i| < Nc, \quad |q - j| < Nc
$$

$Nc$ – size of neighborhood

$\alpha(t)$ -- a positive value decaying as training proceeds
Filter Clustering Example (Burrascano et al)
Other Advanced Structures:

- Knowledge Based Neural Networks embedding application specific knowledge into networks

![Diagram](Empirical formula + Equivalent Circuit + Pure Neural Networks)