Introduction to Neural Networks: Structure and Training

Professor Q.J. Zhang

Department of Electronics

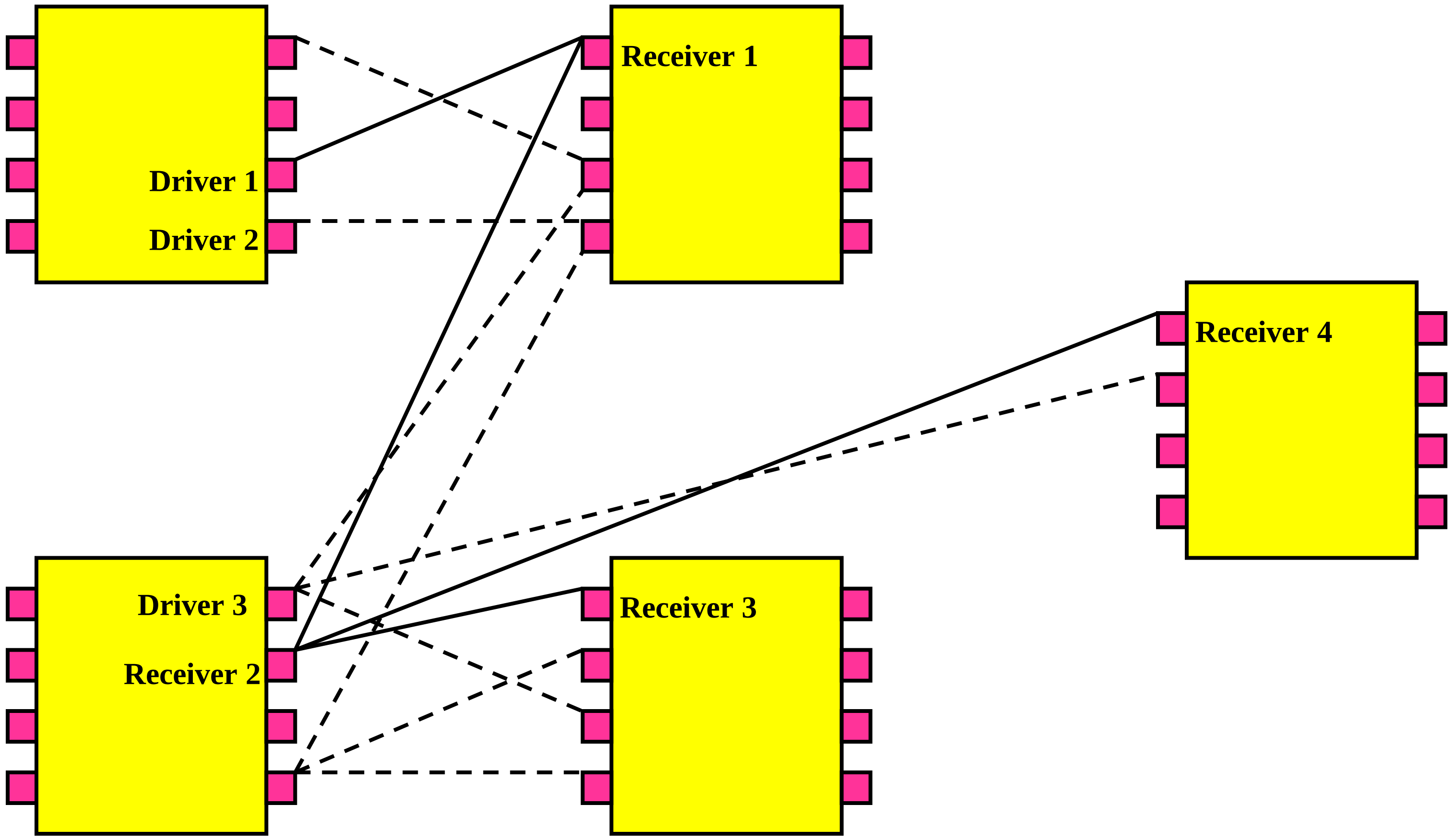
Carleton University, Ottawa, Canada

www.doe.carleton.ca/~qjz, qjz@doe.carleton.ca

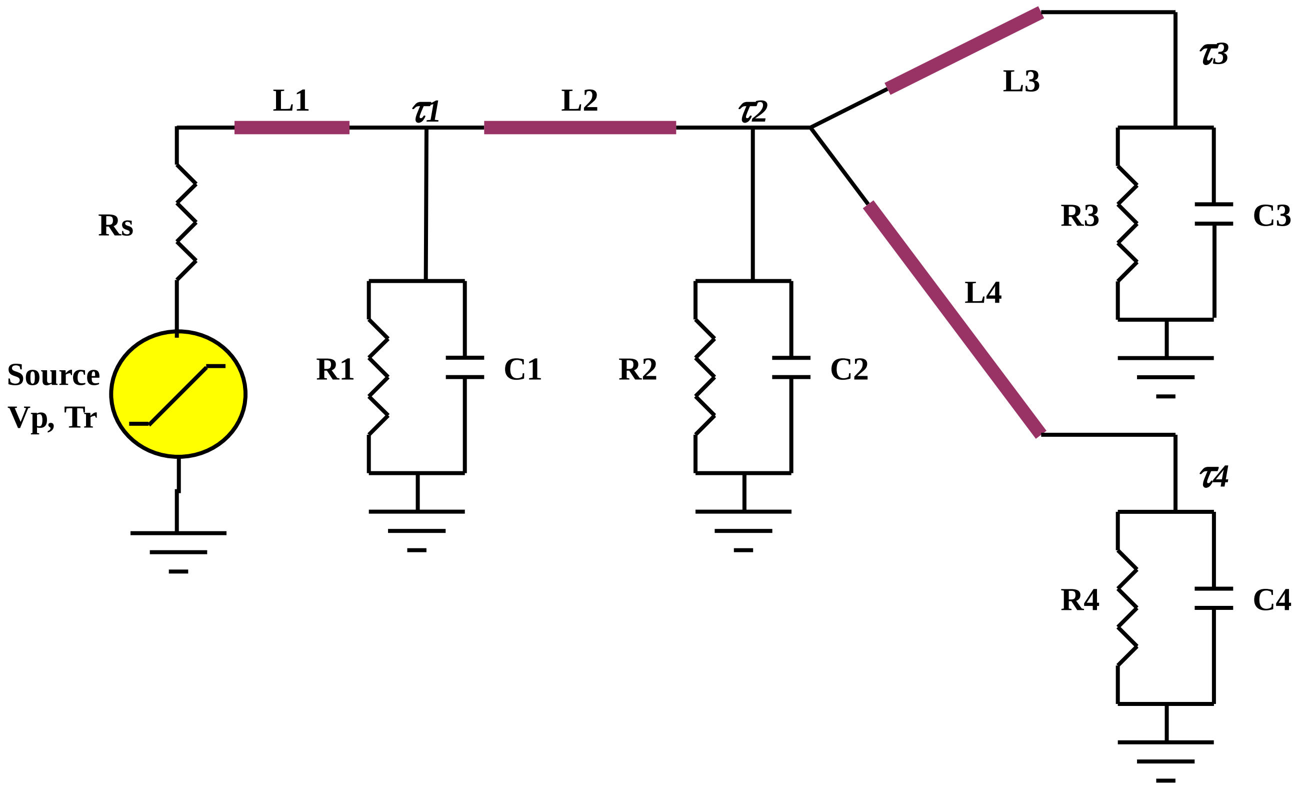
A Quick Illustration Example:

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

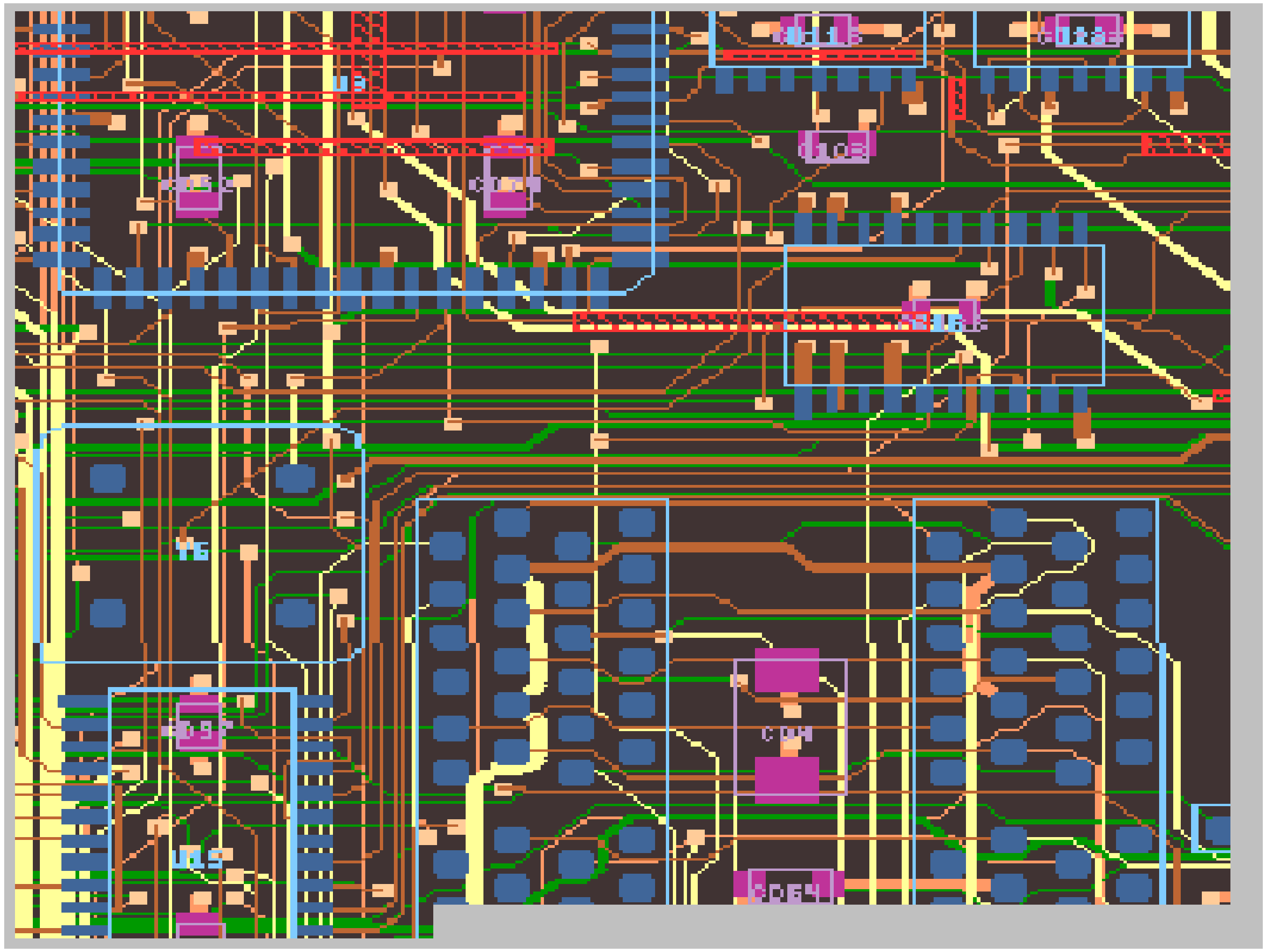
High-Speed VLSI Interconnect Network



Circuit Representation of the Interconnect Network



Massive Analysis of Signal Delay



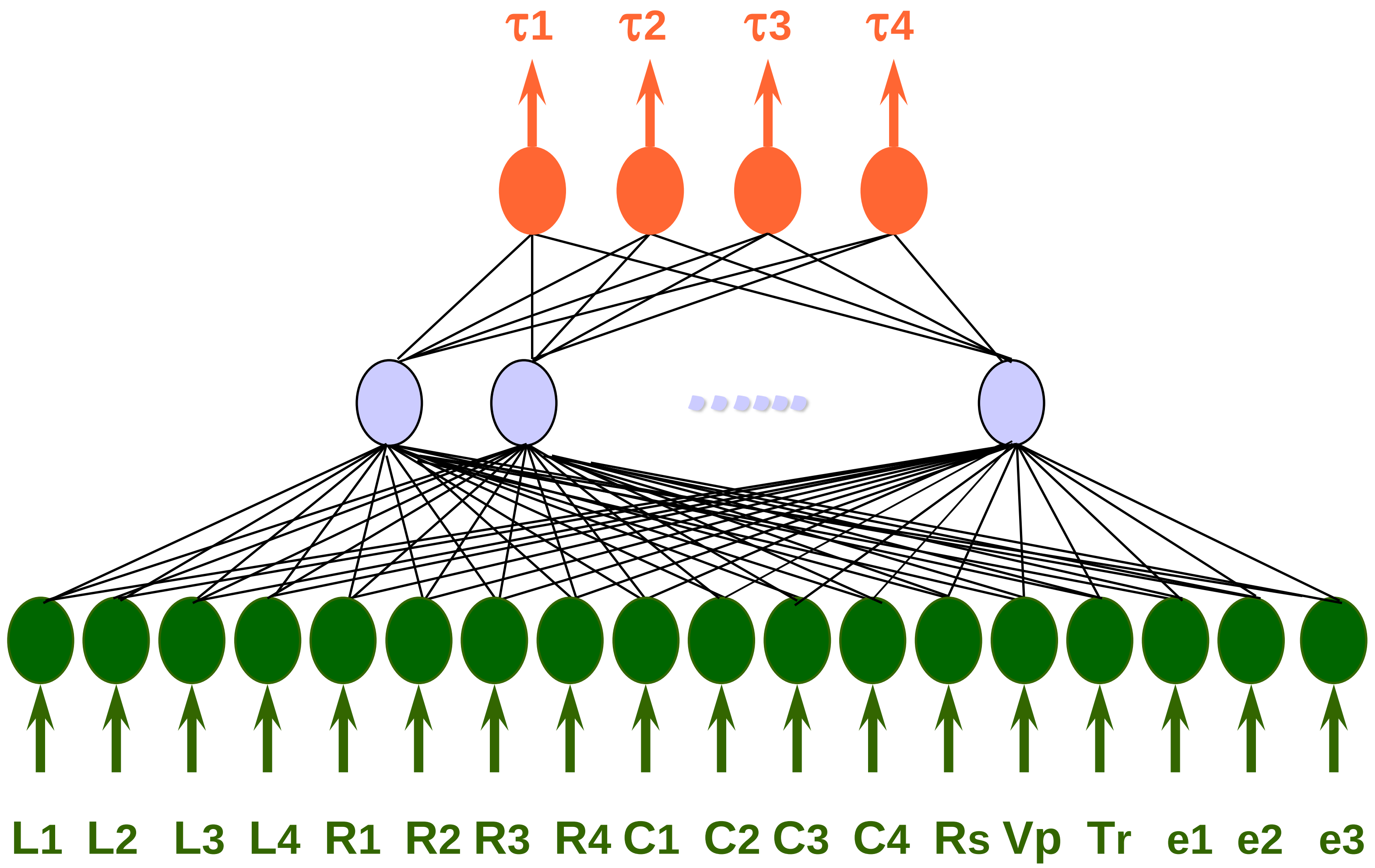
Need for a Neural Network Model

- 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

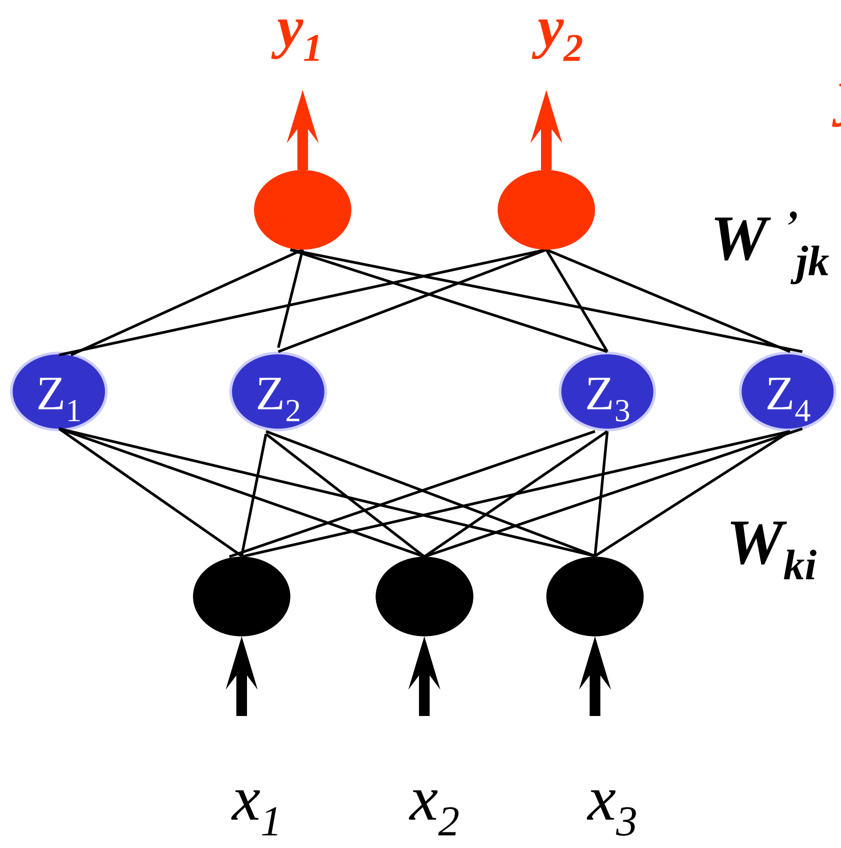
- During PCB design/optimization, the interconnect networks need to be adjusted in terms of interconnect lengths, receiver-pin load characteristics, etc, leading to need of repetitive analysis of interconnect networks

- 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



Outputs

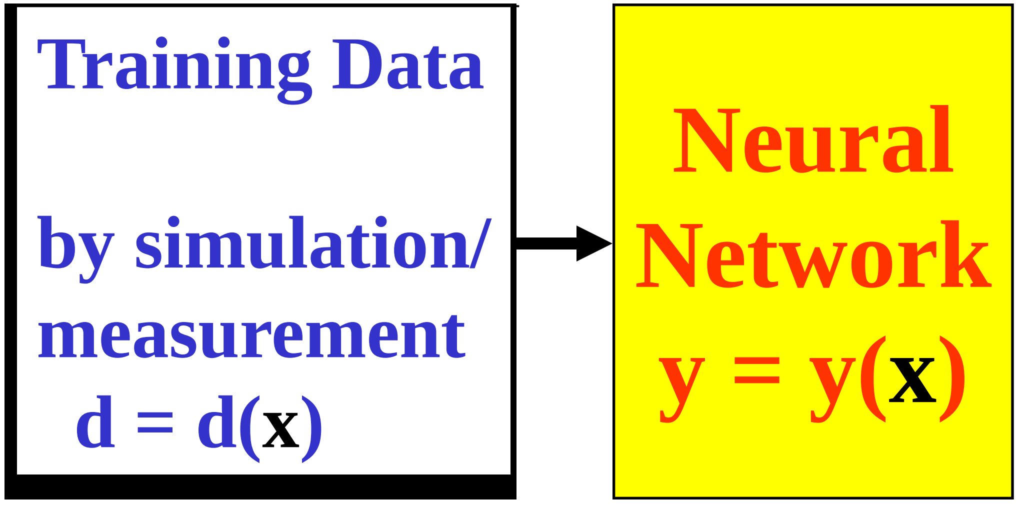
$$y_j = \sum_k W'_{jk} Z_k$$

Hidden Neuron Values

$$Z_k = \tanh\left(\sum_i W_{ki} x_i\right)$$

Inputs

Neural Net Training



Objective:

to adjust W,V such that

$$\min_{W,V} \sum_{x} (y - d)^2$$

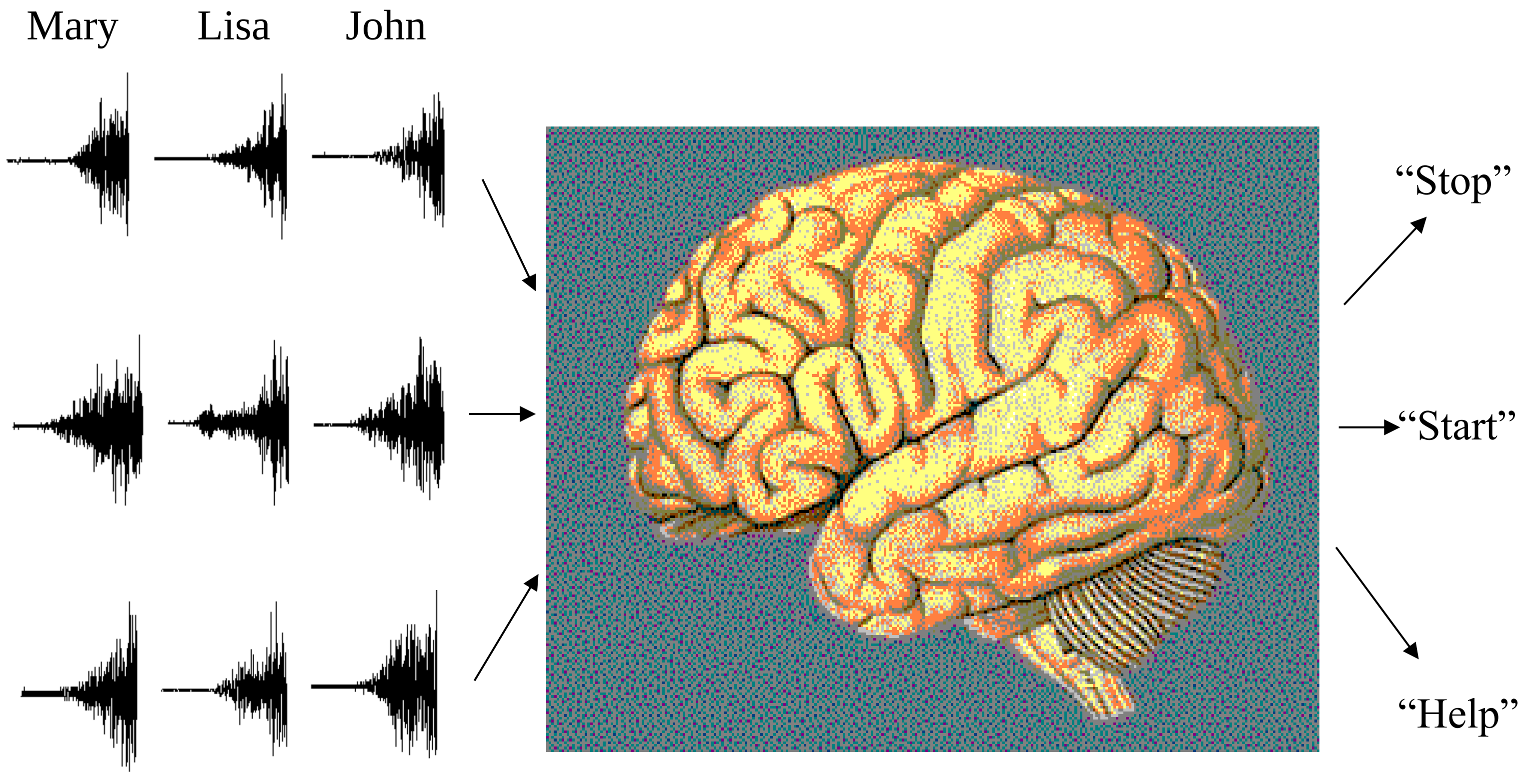
Simulation Time for 20,000 Interconnect Configurations

Method	**CPU**
Circuit Simulator (NILT)	34.43 hours
AWE	9.56 hours
Neural Network Approach	6.67 minutes

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



A Biological Neuron

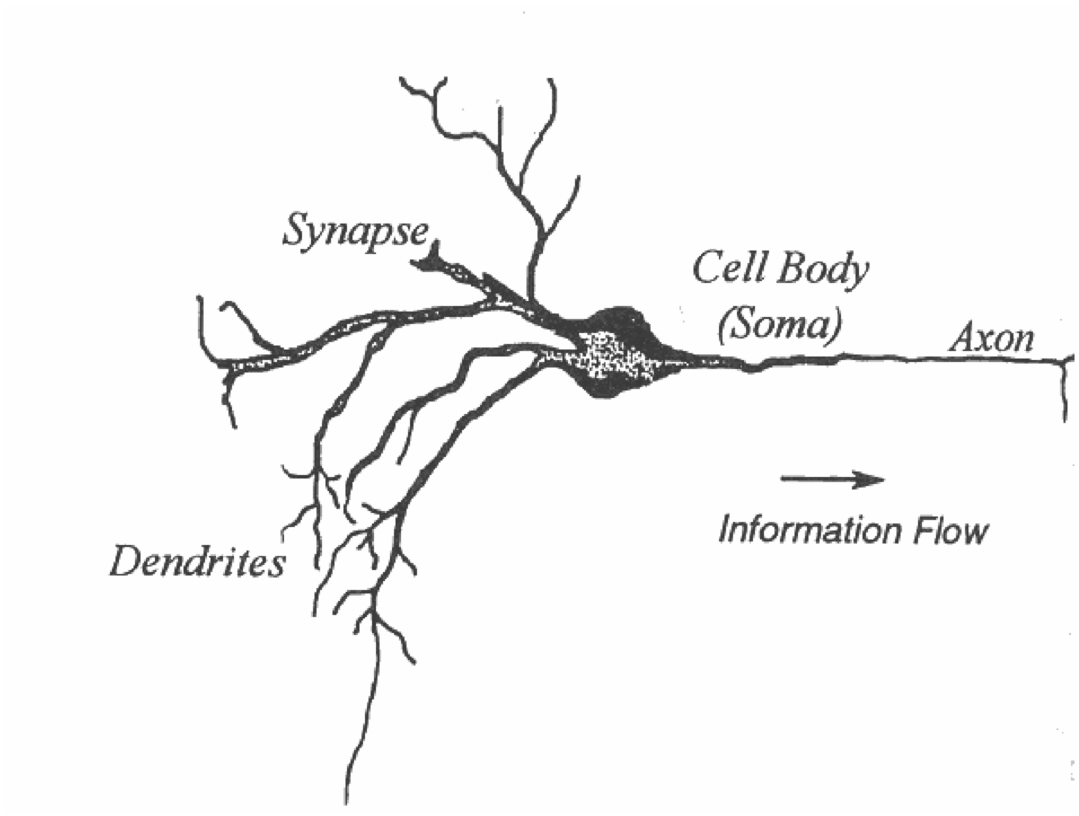


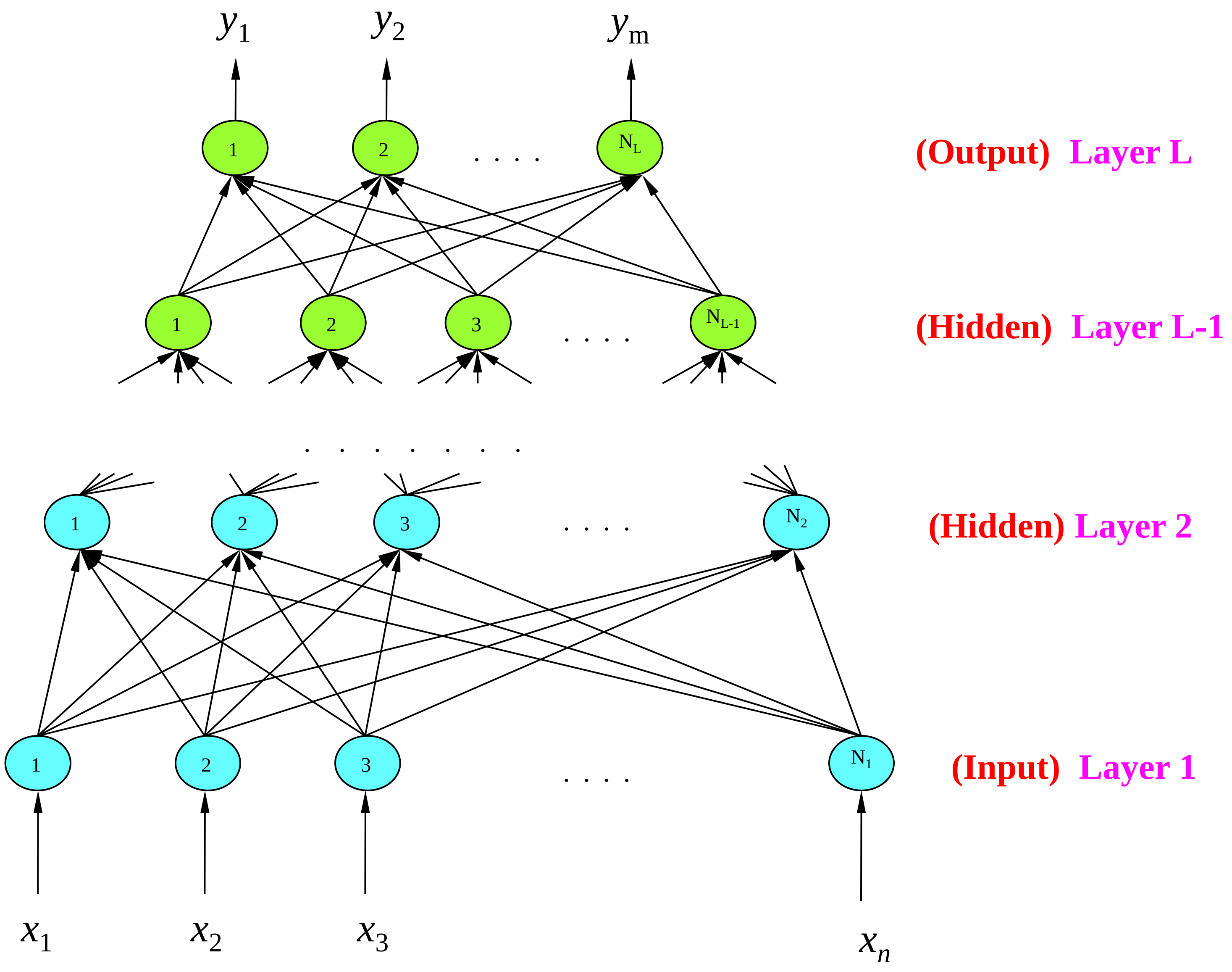
Figure from Reference [L.H. Tsoukalas and R.E. Uhrig, Fuzzy and Neural Approaches in Engineering, Wiley, 1997.]

Neural Network Structures

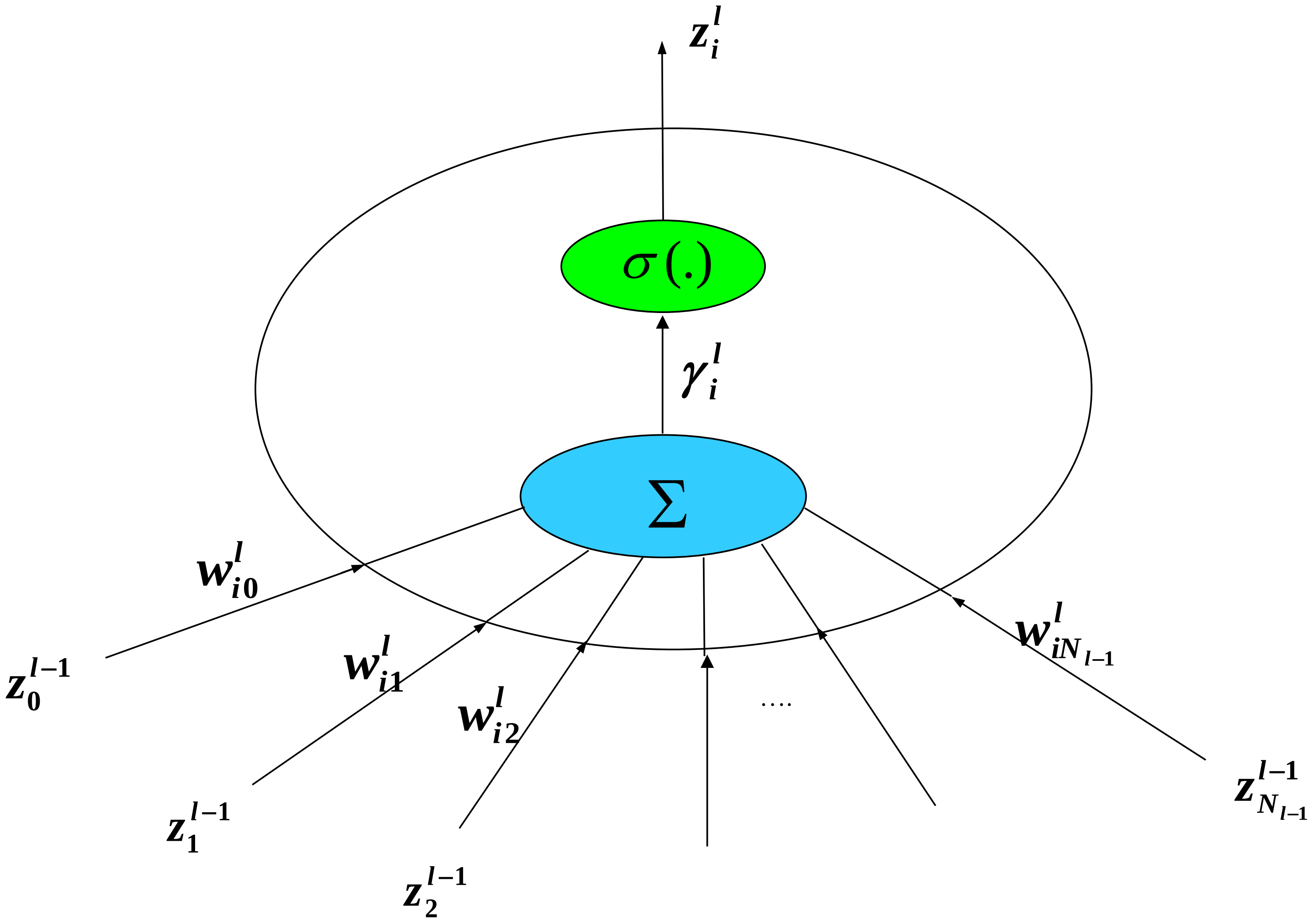
Neural Network Structures

- **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



Information Processing In a Neuron

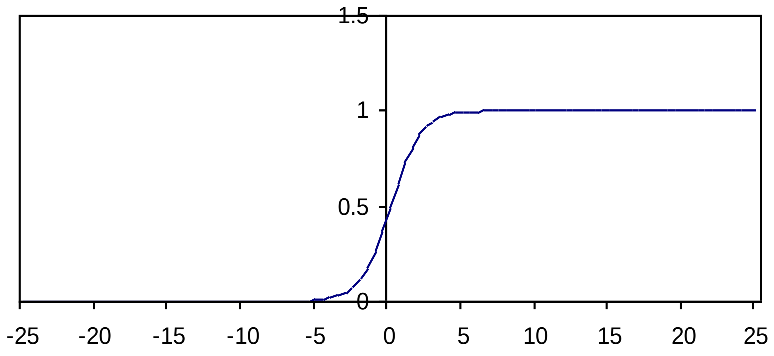


Neuron Activation Functions

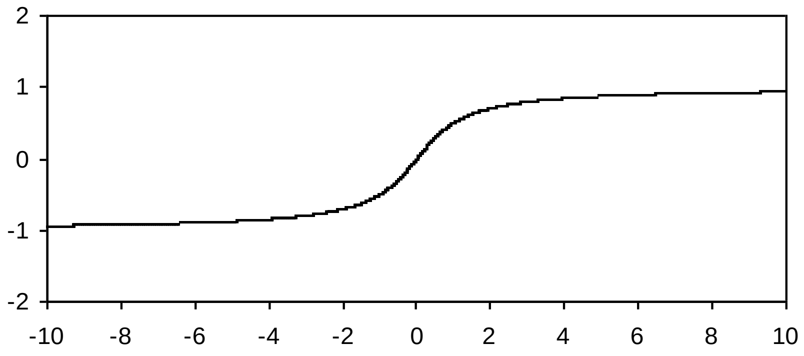
- **Input layer neurons simply relay the external inputs to the neural network**
- **Hidden layer neurons have smooth switch-type activation functions**
- **Output layer neurons can have simple linear activation functions**

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

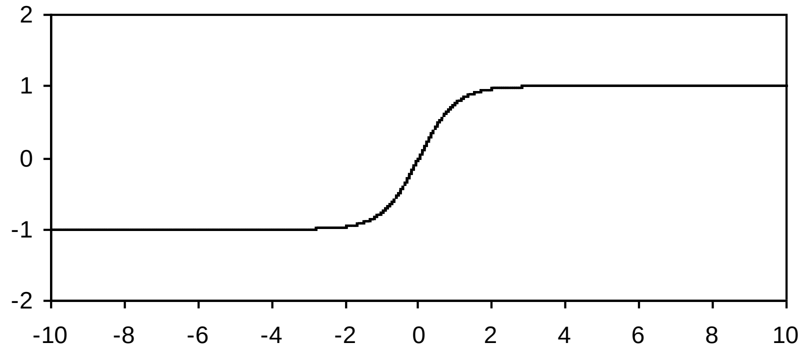
Sigmoid



Arctangent



Hyperbolic tangent



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

- **Sigmoid function:**

$$z = \sigma(\gamma) = \frac{1}{1 + e^{-\gamma}}$$

- **Arctangent function:**

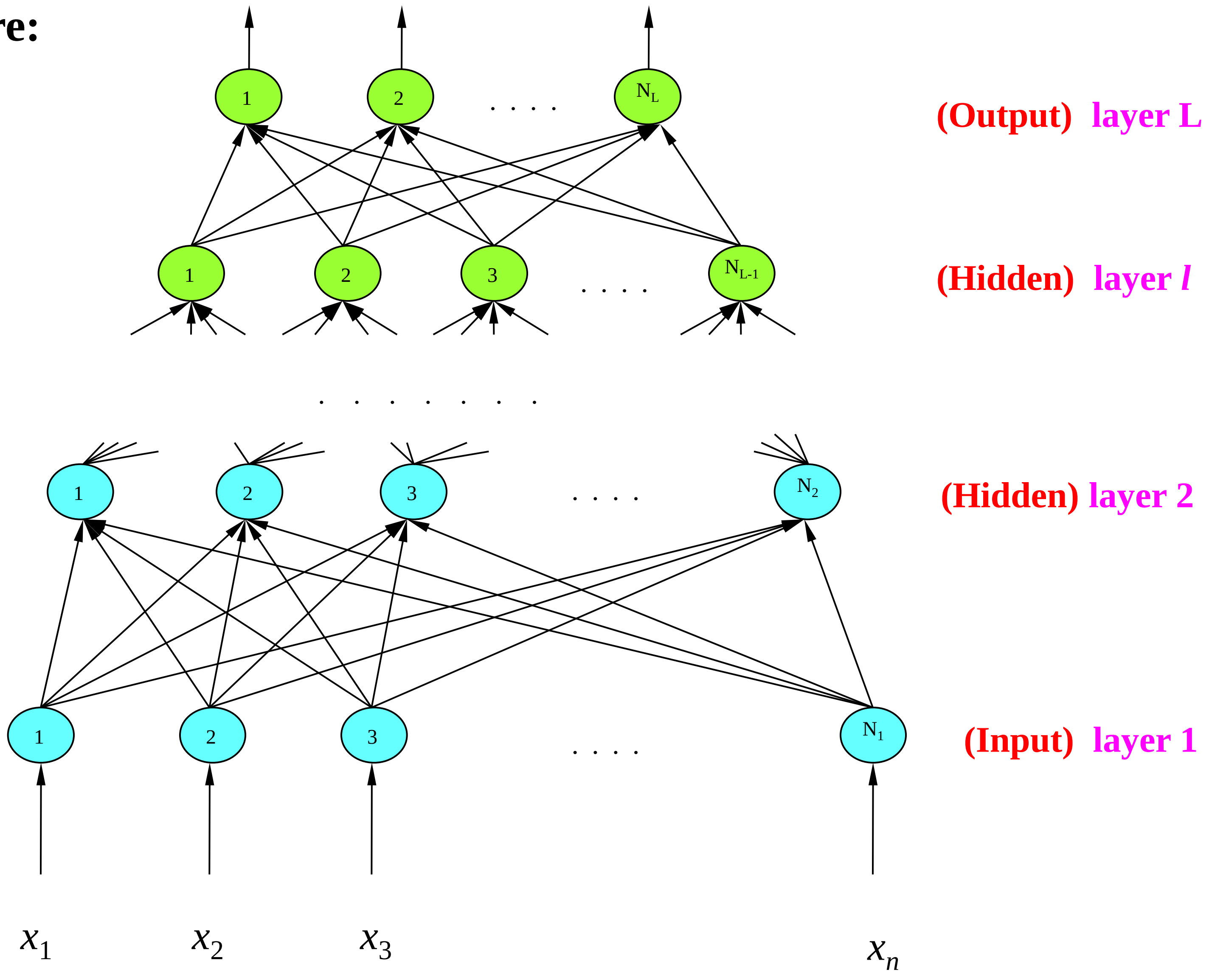
$$z = \sigma(\gamma) = \frac{2}{\pi} \arctan(\gamma)$$

- **Hyperbolic Tangent function:**

$$z = \sigma(\gamma) = \frac{e^{+\gamma} - e^{-\gamma}}{e^{+\gamma} + e^{-\gamma}}$$

Multilayer Perceptrons (MLP):

Structure:



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 = $x_1, x_2, \cdots, x_n$ (where $n = N_1$)

NN outputs = $y_1, y_2, \cdots, y_m$ (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 to layer #1, and feed outputs of layer #*l*-1 to *l*.

For $l = 1: \quad z_i^{\,l} = x_i, \quad i = 1,2,\cdots,n, \quad n = N_1$

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

and solution is $y_i = z_i^L, \quad i = 1,2,\cdots,m, \quad m = N_L$

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 $\sigma(.)$ be any continuous sigmoid function, the finite sums of the form:

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

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

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

where:

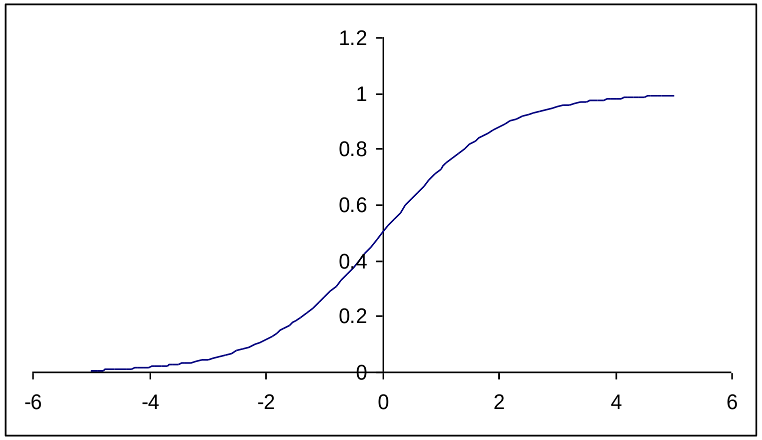
I_n -- n-dimensional unit cube $[0,1]^n$

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

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

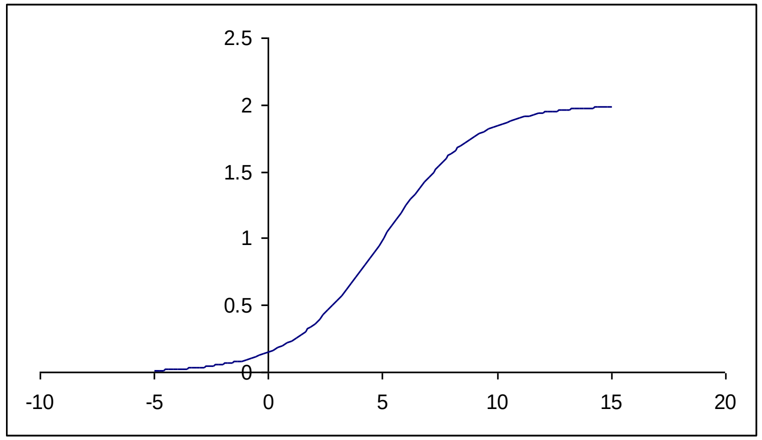
**e.g., Original problem is $y = f(x)$, where $f \in C(I_n)$,
the form of $\bar{f}(x)$ is a 3-layer-perceptron NN.**

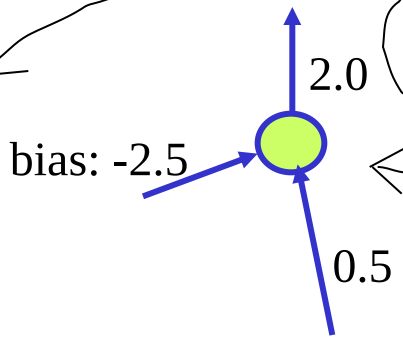
Illustration of the Effect of Neural Network Weights



Standard signmoid function

$1/(1 + e^{-x})$

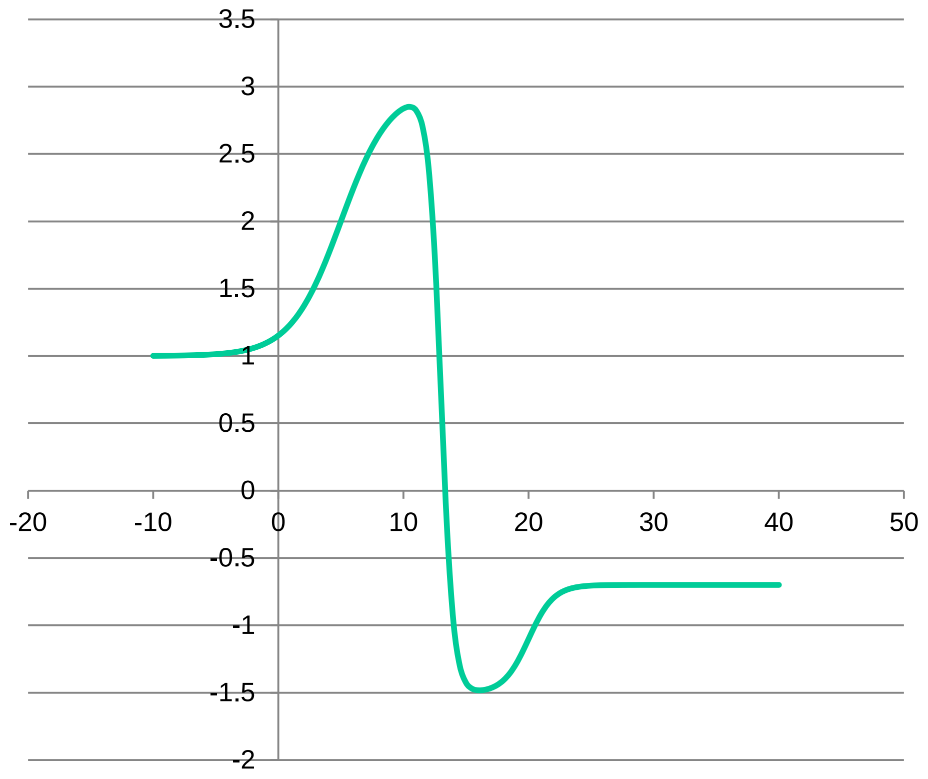




Hidden neuron
$1/(1 + e^{-(0.5x - 2.5)})$

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.

Illustration of the Effect of Neural Network Weights - Example with 1 input



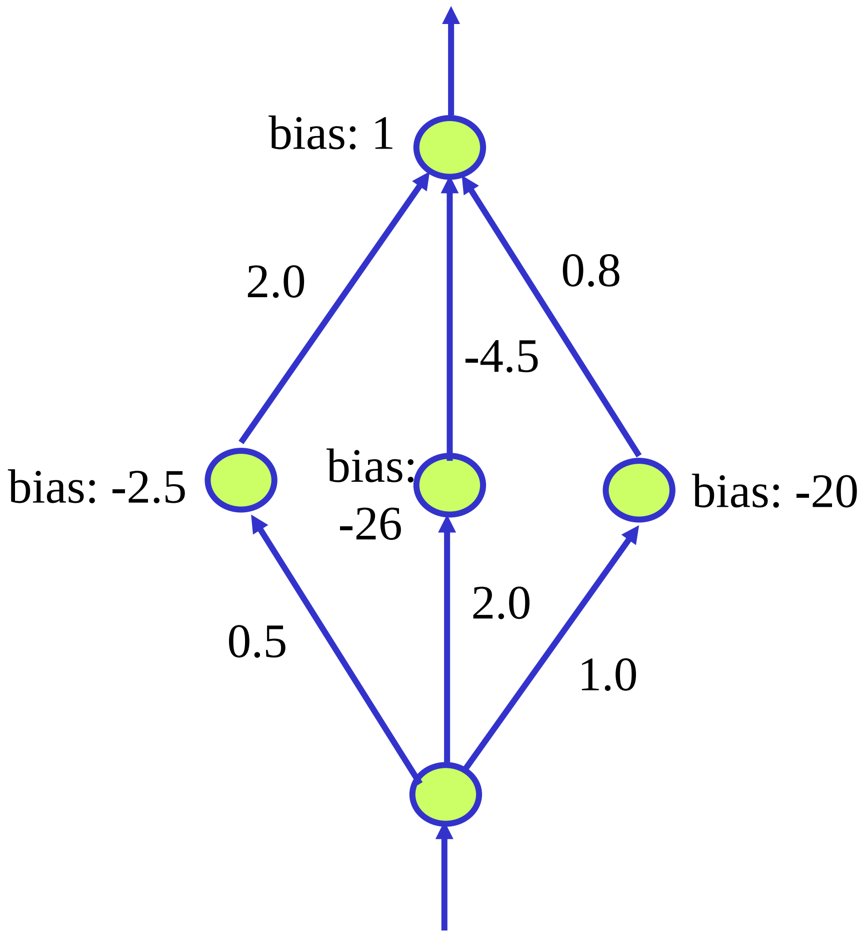
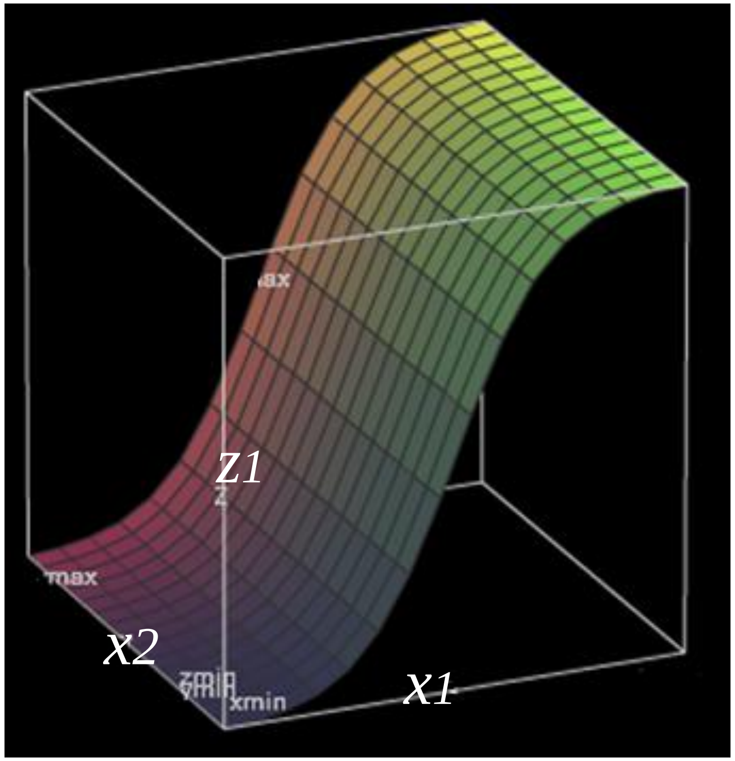
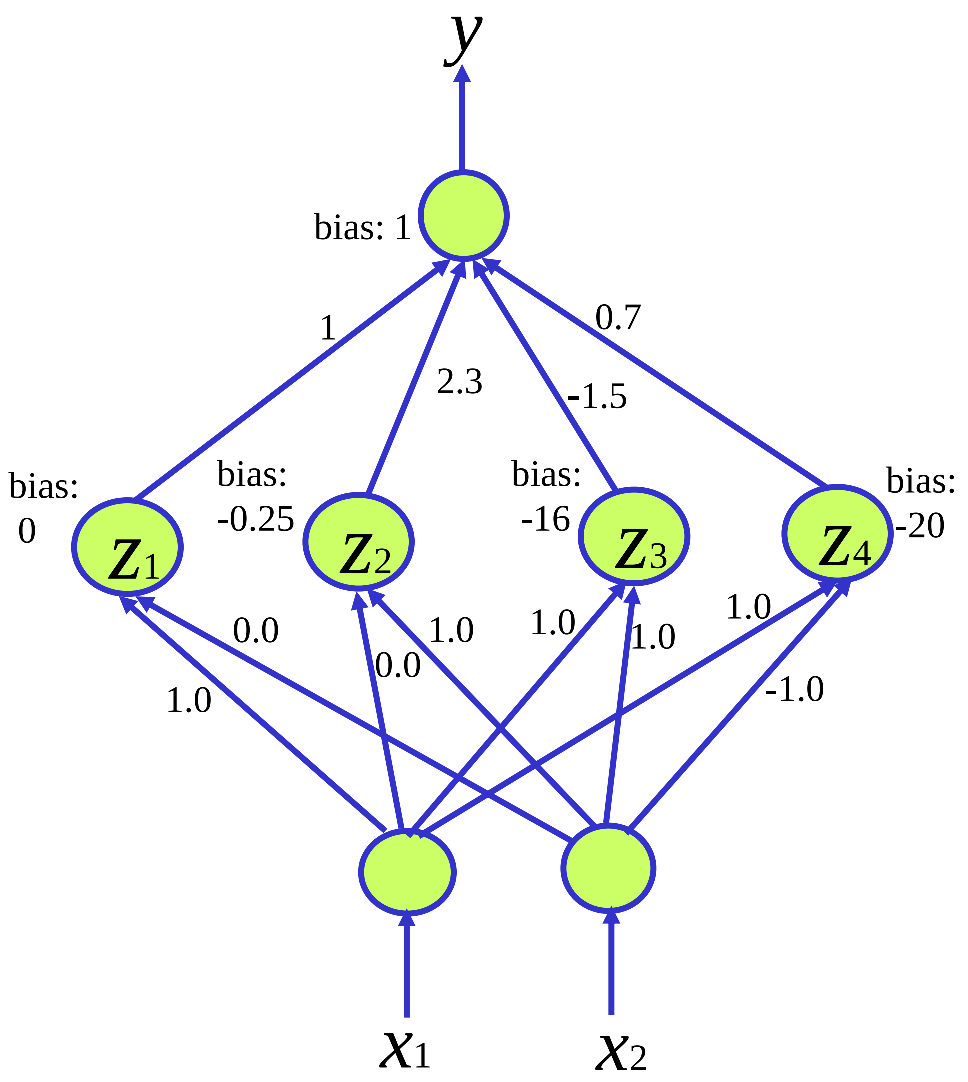


Illustration of the Effect of Neural Network Weights - Example with 2 inputs



z_1 as a function of x_1 and x_2 :

$$z_1 = 1/(1+\exp(-x_1))$$



Q.J. Zhang, Carleton University

Illustration of the Effect of Neural Network Weights - Example with 2 inputs

x1	x2	z1	z2	z3	z4
-1	-1	-1	-1	-1	0
+1	-1	+1	-1	0	0
-1	+1	-1	+1	0	0
+1	+1	+1	+1	+1	0

Assuming arctan activation function

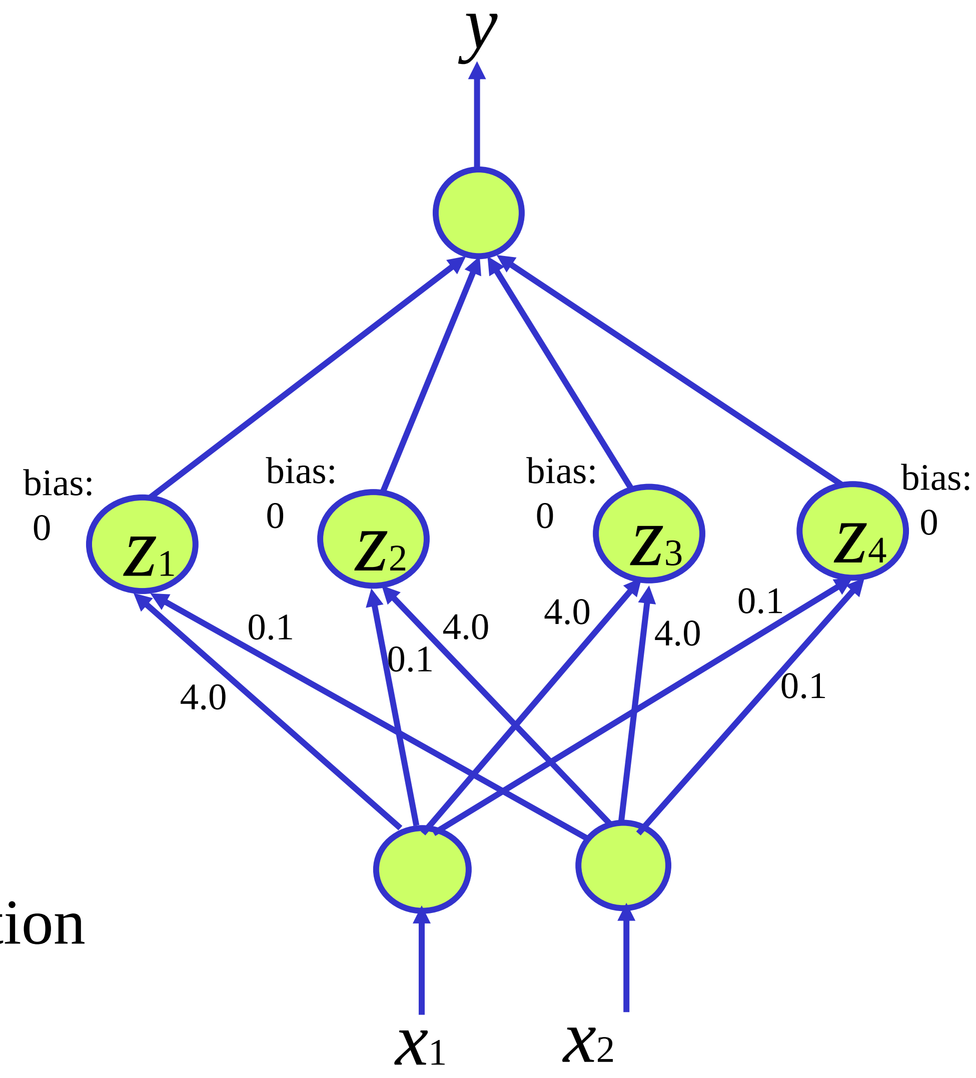
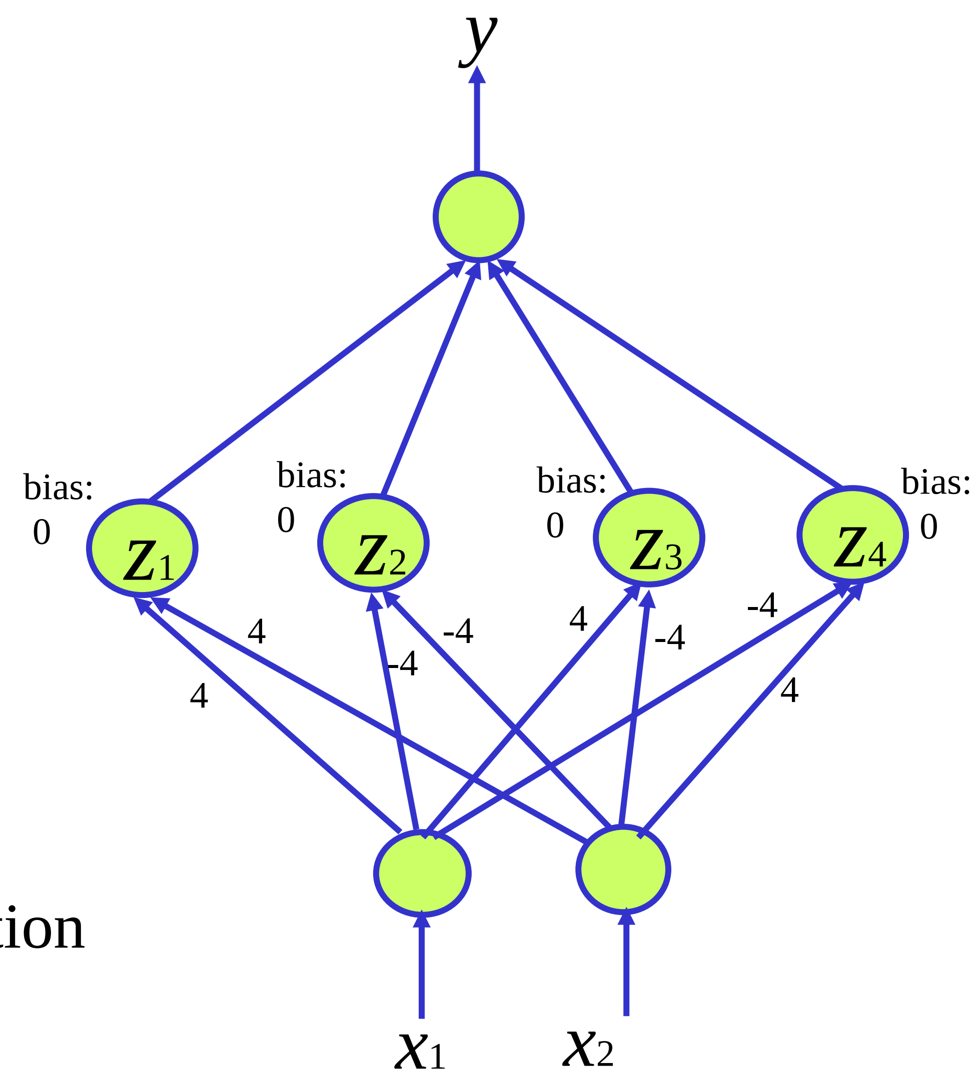


Illustration of the Effect of Neural Network Weights - Example with 2 inputs

x1	x2	z1	z2	z3	z4
-1	-1	-1	+1	0	0
+1	-1	0	0	+1	-1
-1	+1	0	0	-1	+1
+1	+1	+1	-1	0	0

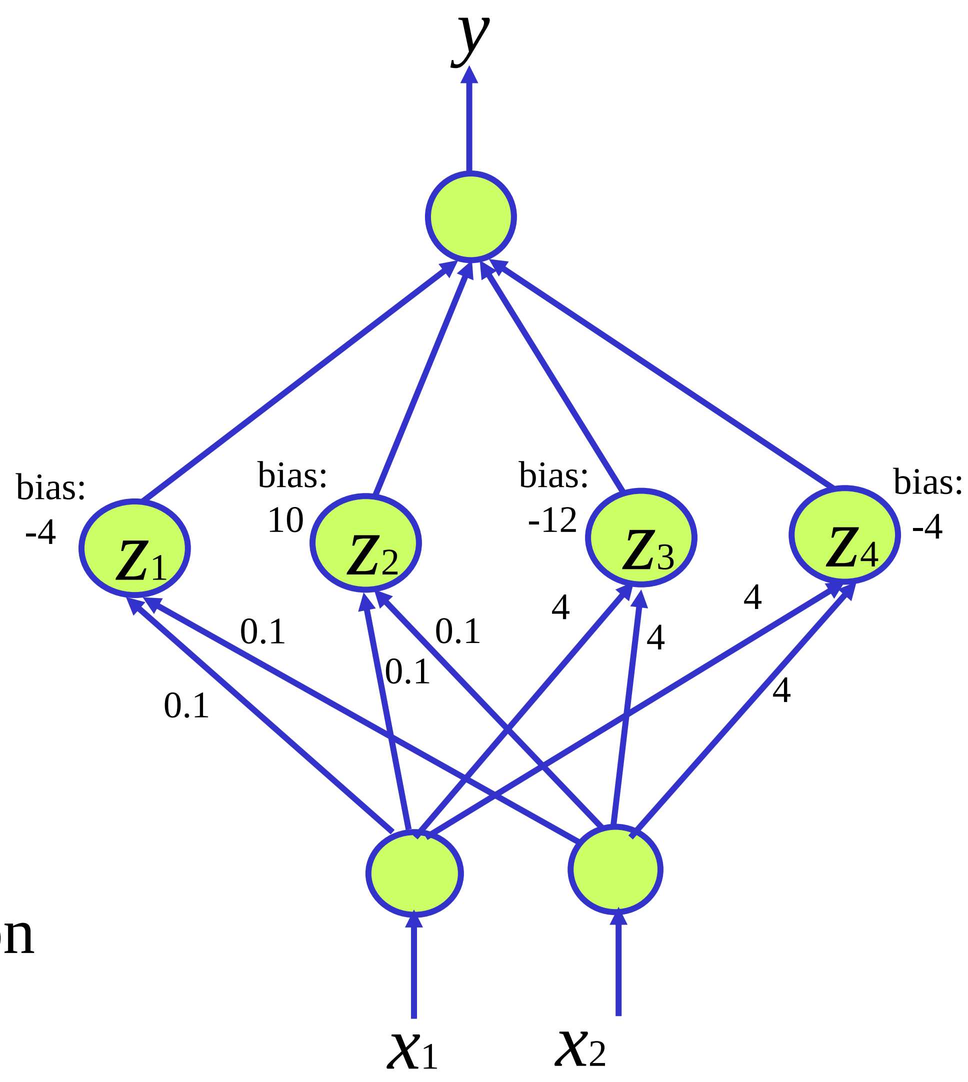


Assuming arctan activation function

Illustration of the Effect of Neural Network Bias Parameters - Example with 2 inputs

x1	x2	z1	z2	z3	z4
-1	-1	-1	+1	-1	-1
-1	+1	-1	+1	-1	-1
+1	-1	-1	+1	-1	-1
+1	+1	-1	+1	-1	+1
40	40	+1	+1	+1	+1
-70	-70	-1	-1	-1	-1

Assuming arctan activation function

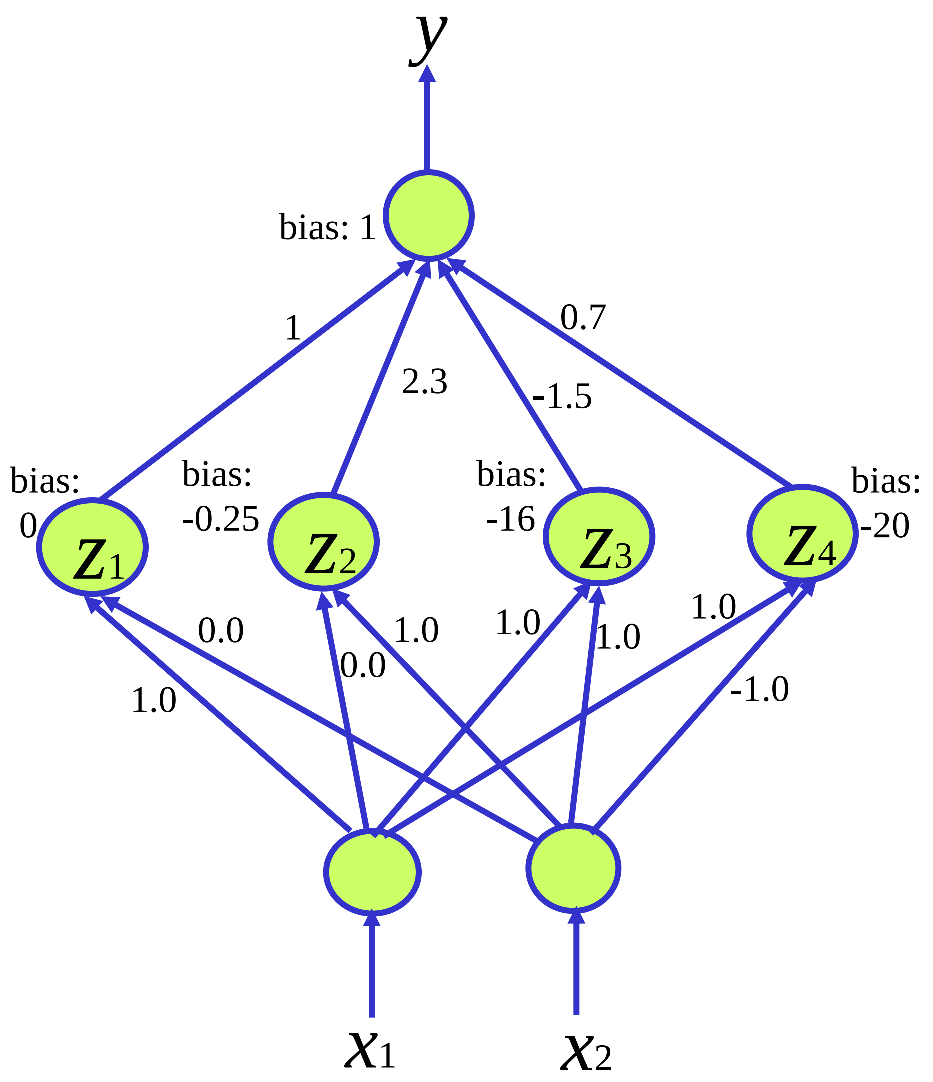


Effect of Neural Network Weights, and Inputs on Neural Network Outputs

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***.

$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{w})$



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

$$\textbf{Solution:}\ \begin{cases}\textbf{experience}\\ \textbf{trial/error}\\ \textbf{adaptive schemes}\end{cases}$$

Question: How many layers are needed?

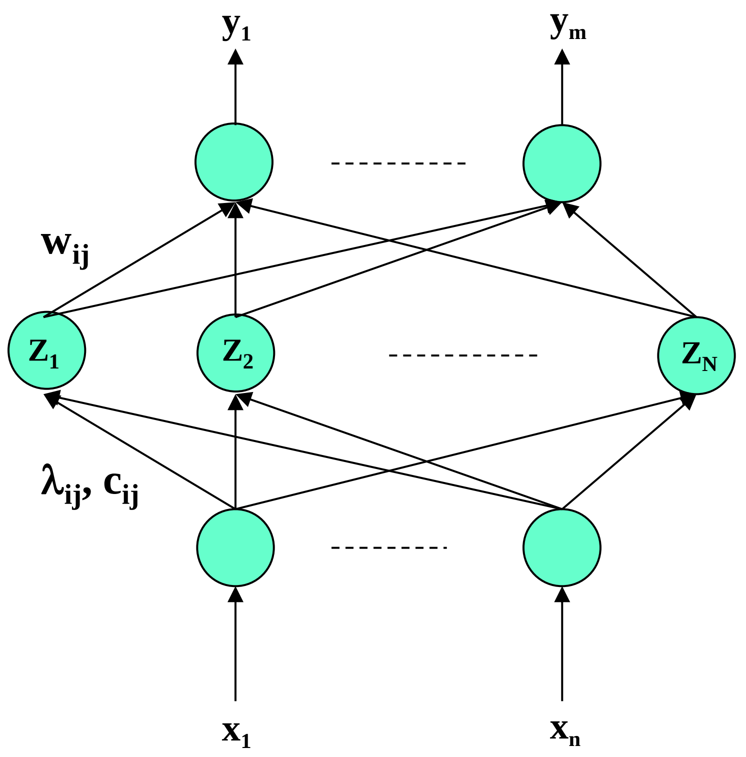
- 3 or more layers are necessary and sufficient for arbitrary nonlinear approximation

- 3 or 4 layers are used more frequently

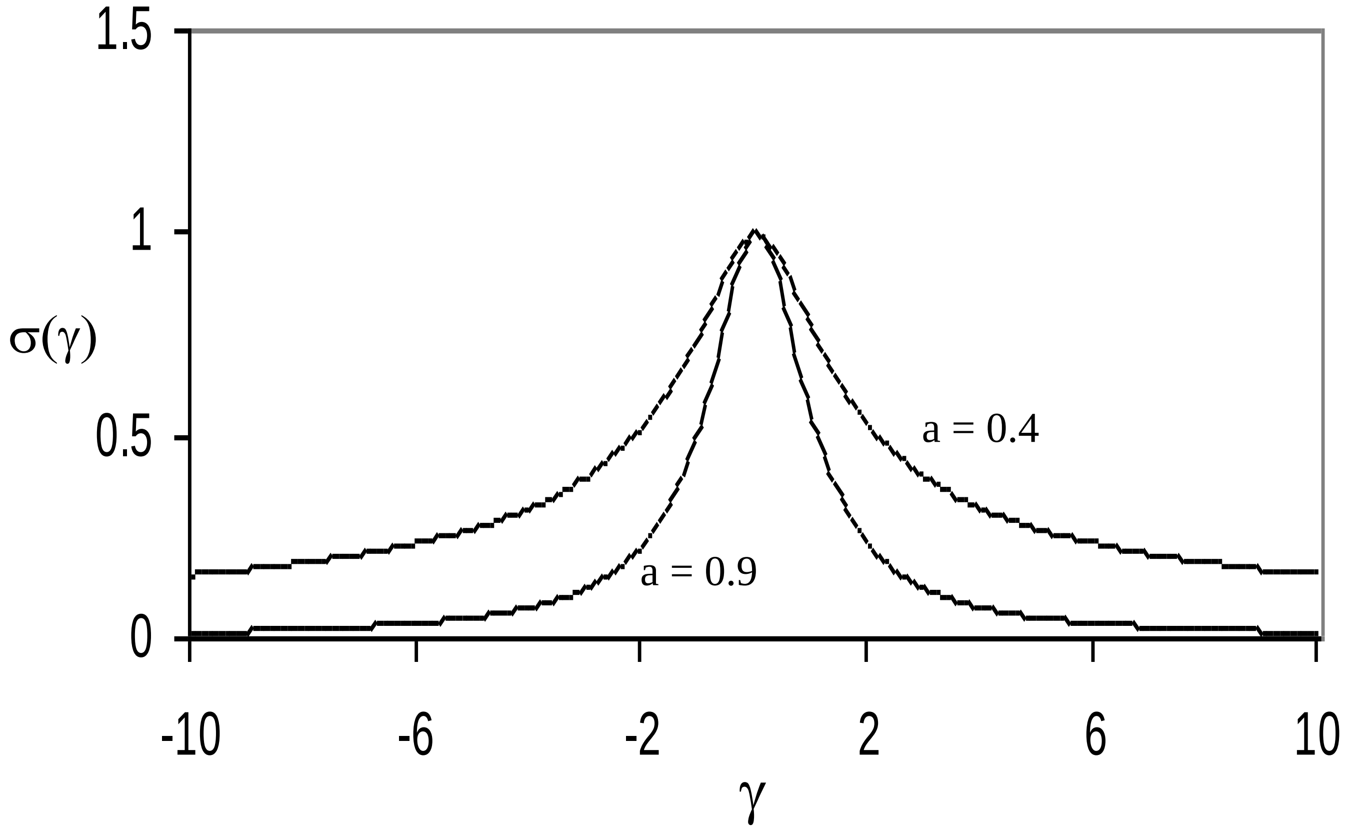
- more layers allow more effective representation of hierarchical information in the original problem

Radial Basis Function Network (RBF)

Structure:

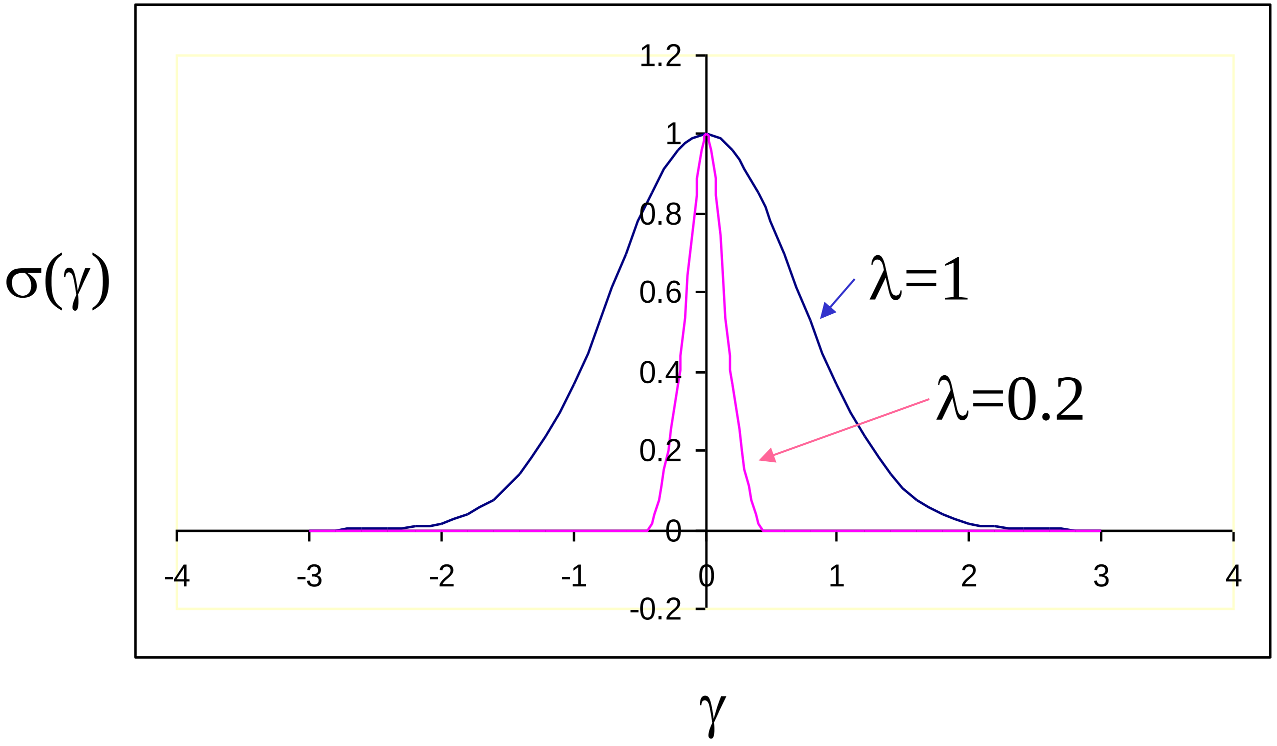


RBF Function (multi-quadratic)



$$\sigma(\gamma) = \frac{1}{(c^2 + \gamma^2)^{\alpha}}, \ \alpha > 0$$

RBF Function (Gaussian)



$$\sigma(\gamma) = exp(-(\gamma/\lambda)^2)$$

RBF Feedforward:

$$y_k = \sum_{i=0}^{N} w_{ki} z_i \qquad k = 1, 2, \cdots, m$$

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

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

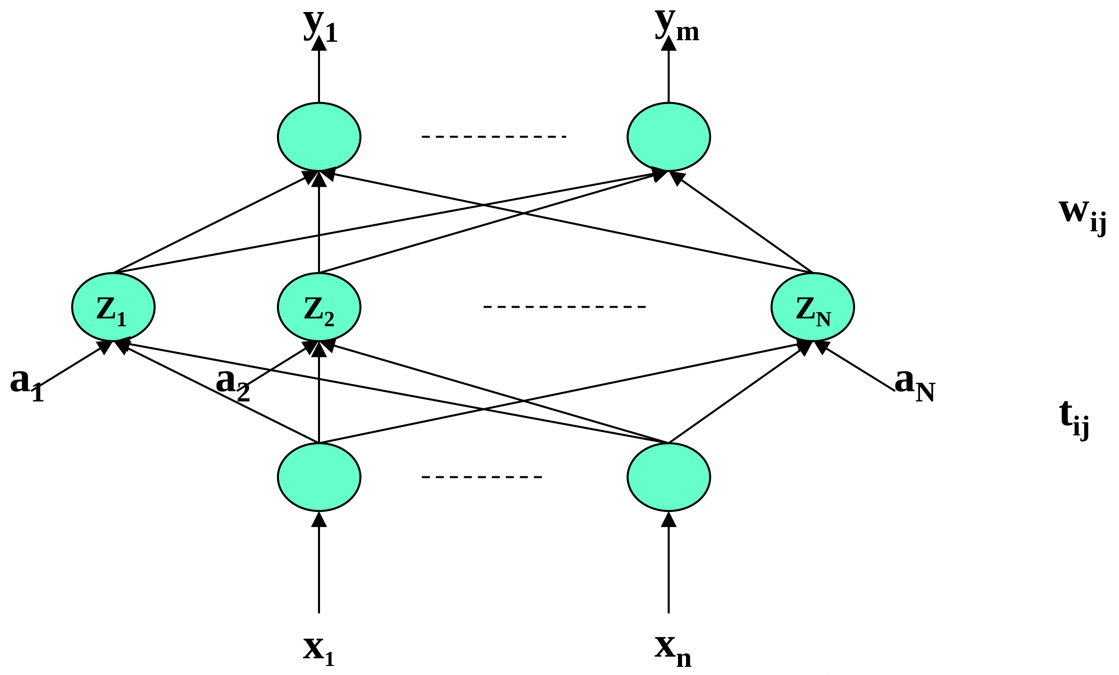
c_{ij} is the center of radial basis function, λ_{ij} is the width factor.

So a set of parameters c_{ij} (*i* = *1, 2, …, N*, *j* = *1, …, n*) represent centers of the RBF. Parameters λ_{ij}, (*i* = *1, 2, …, N*, *j* = *1, …, 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:

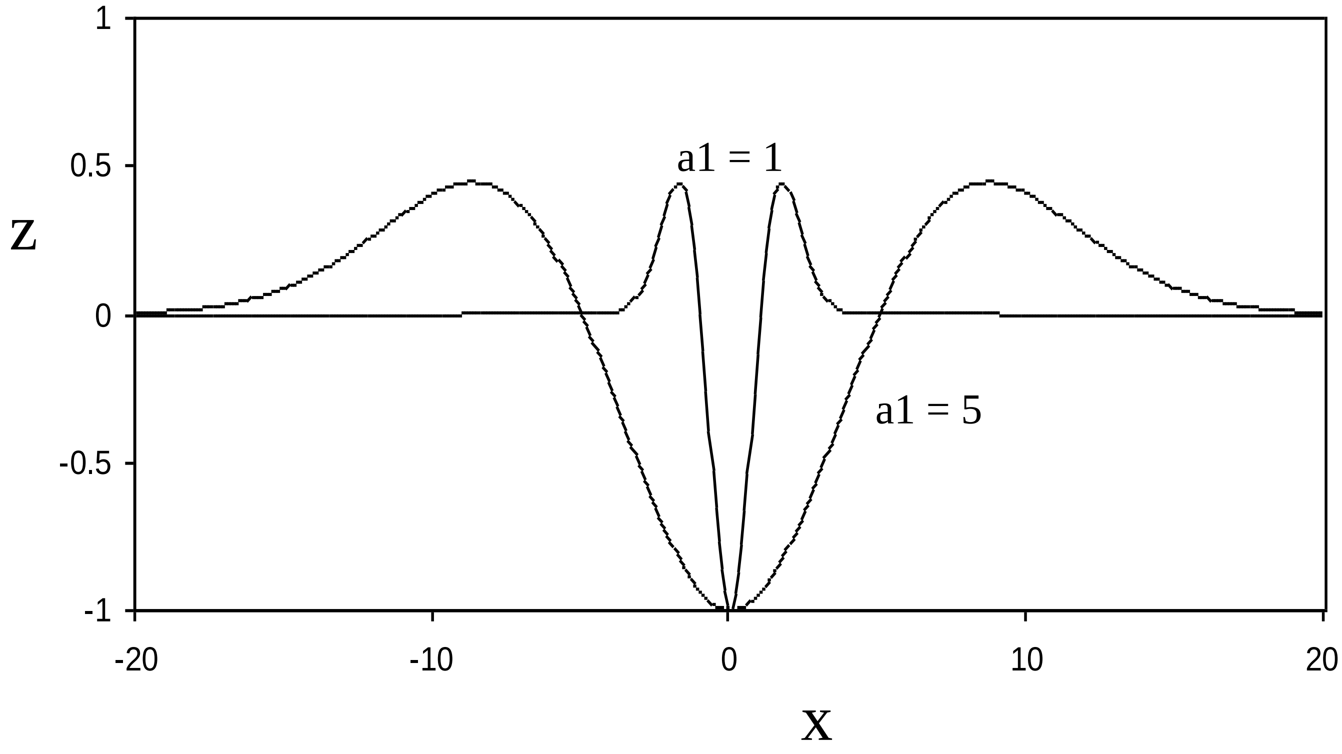


For hidden neuron z_j: $$z_j = \sigma(\gamma_j) = \psi\left(\frac{x - t_j}{a_j}\right)$$

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:

$$\psi(\gamma_j) = \sigma(\gamma_j) = (\gamma_j^2 - n)e^{-\frac{\gamma_j^2}{2}}$$ **where** $$\gamma_j = \left\| \frac{x - t_j}{a_j} \right\| = \sqrt{\sum_{i=1}^{n} \left(\frac{x_i - t_{ji}}{a_j} \right)^2}$$

Wavelet Function



Wavelet Transform:

R – space of a real variable

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

A function $f: R^n \rightarrow R$ is radial, if a function $g: R \rightarrow R$, 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

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

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:

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

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: $\leq$

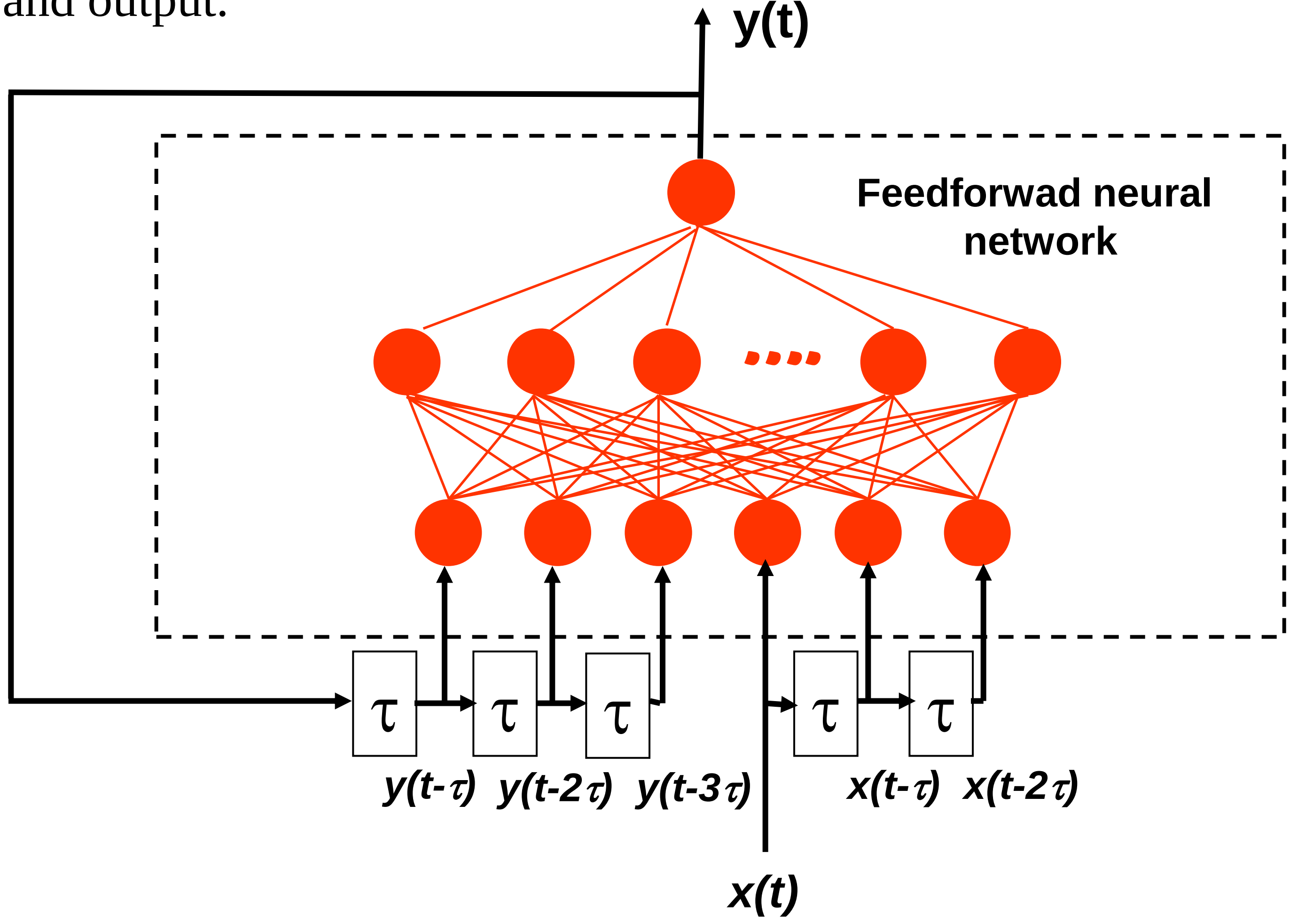
Multilayer Perceptrons (MLP)

Radial Basis Function Networks (RBF)

Wavelet Networks

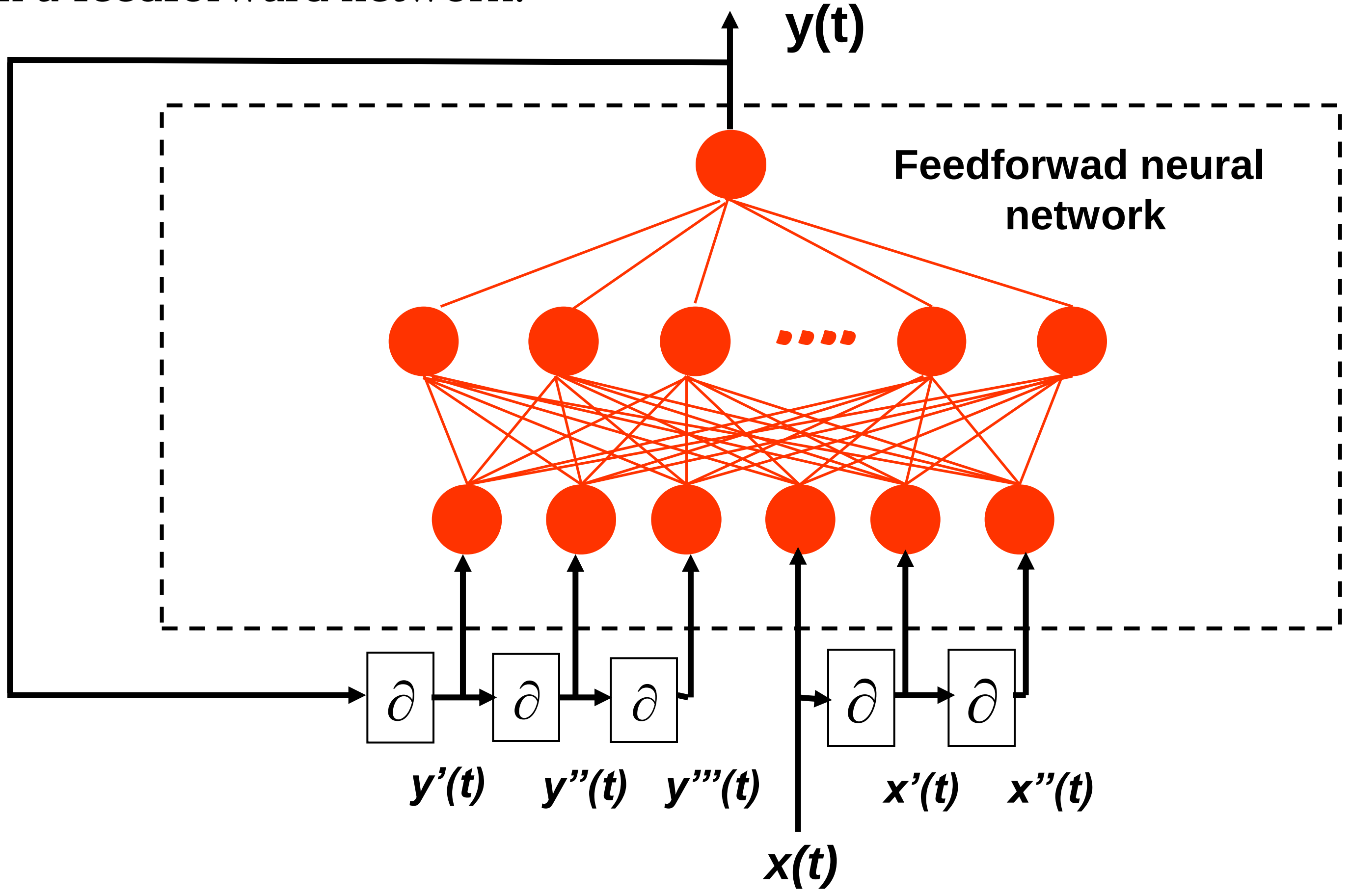
Recurrent Neural Network (RNN): Discrete Time Domain

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, .., 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:

$$c_i = \frac{1}{size(R_i)} \sum_{k \in R_i} x_k$$

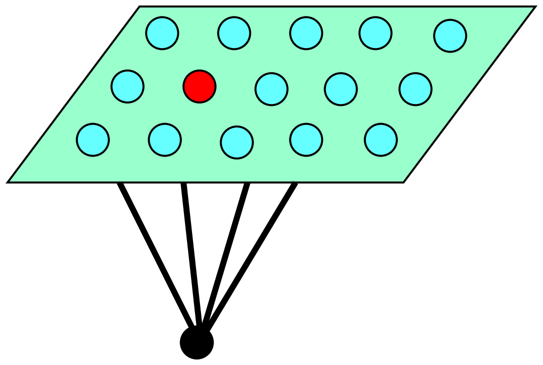
and the process continues until c_i does not move any more.

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



input

Two-dimensional array of neurons

Training of SOM:

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

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

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

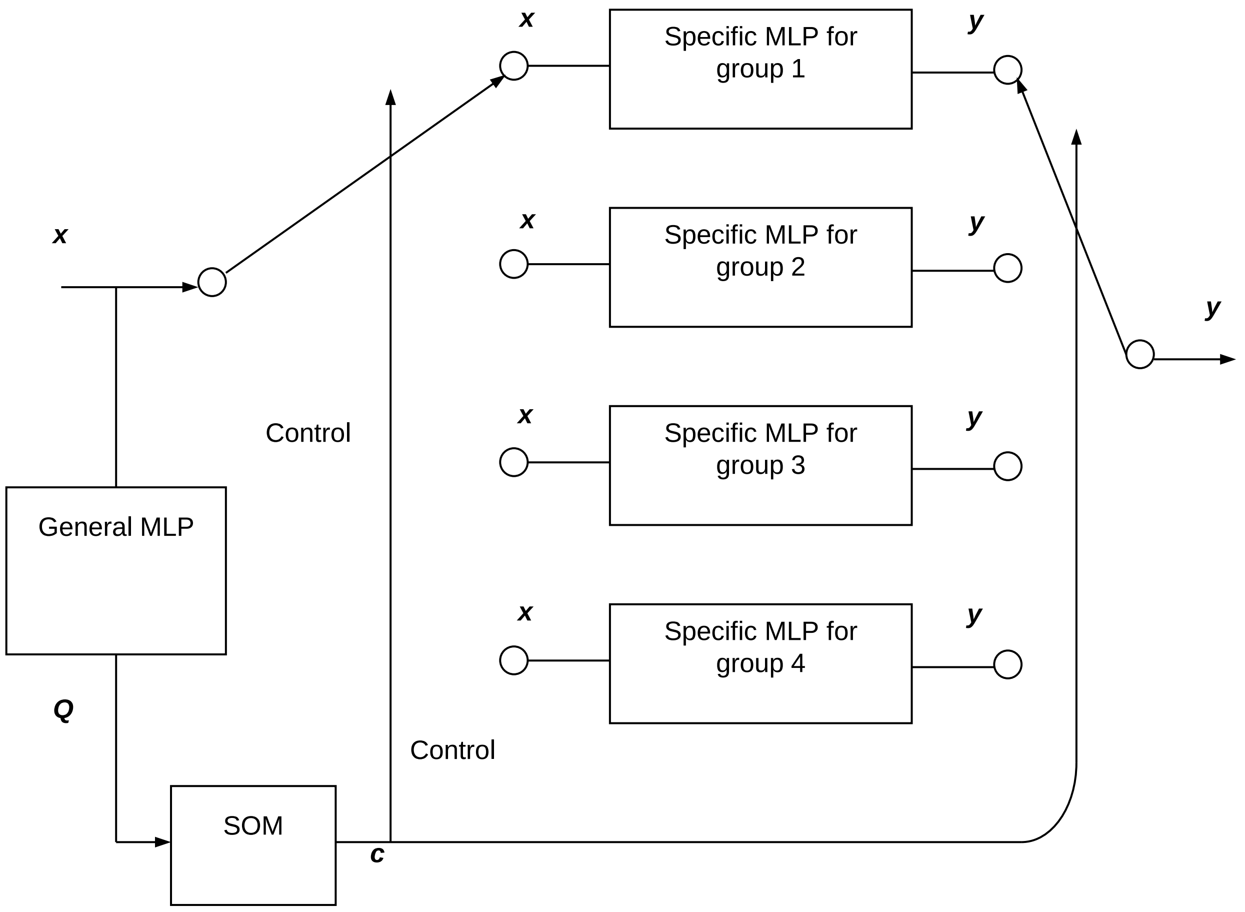
$$c_{pq} = c_{pq} + \alpha(t)(x_k - c_{pq})$$

where $|p - i| < Nc$, $|q - j| < Nc$

Nc **– size of neighborhood**

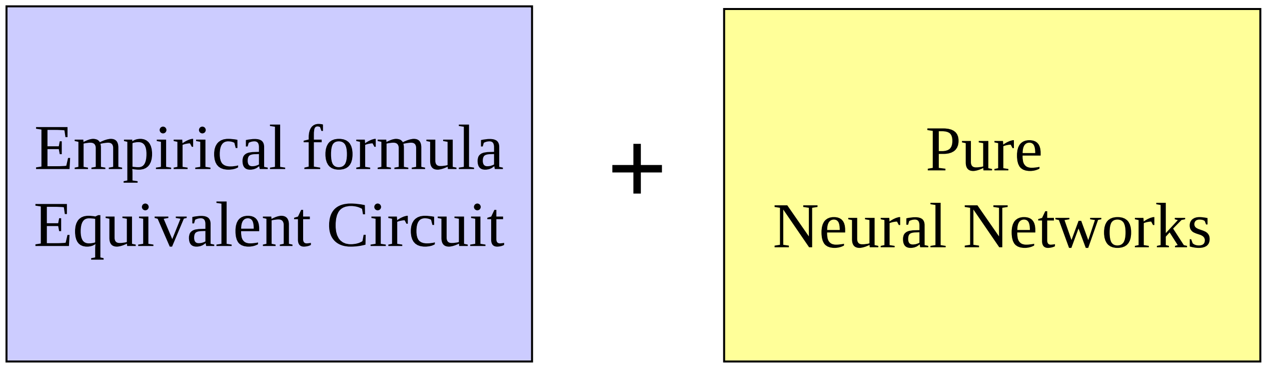
$\alpha(t)$ **-- a positive value decaying as training proceeds**

Filter Clustering Example (Burrascano et al)



Other Advanced Structures:

- **Knowledge Based Neural Networks embedding application specific knowledge into networks**



Other Advanced Structures:

- Ensemble Neural Networks:

  split the training data into multiple sets (such that each set of data still cover the entire training region)

  train multiple neural networks

  combine the multiple neural networks to form the overall neural network

  The quality of the overall neural network out-performs individual neural networks