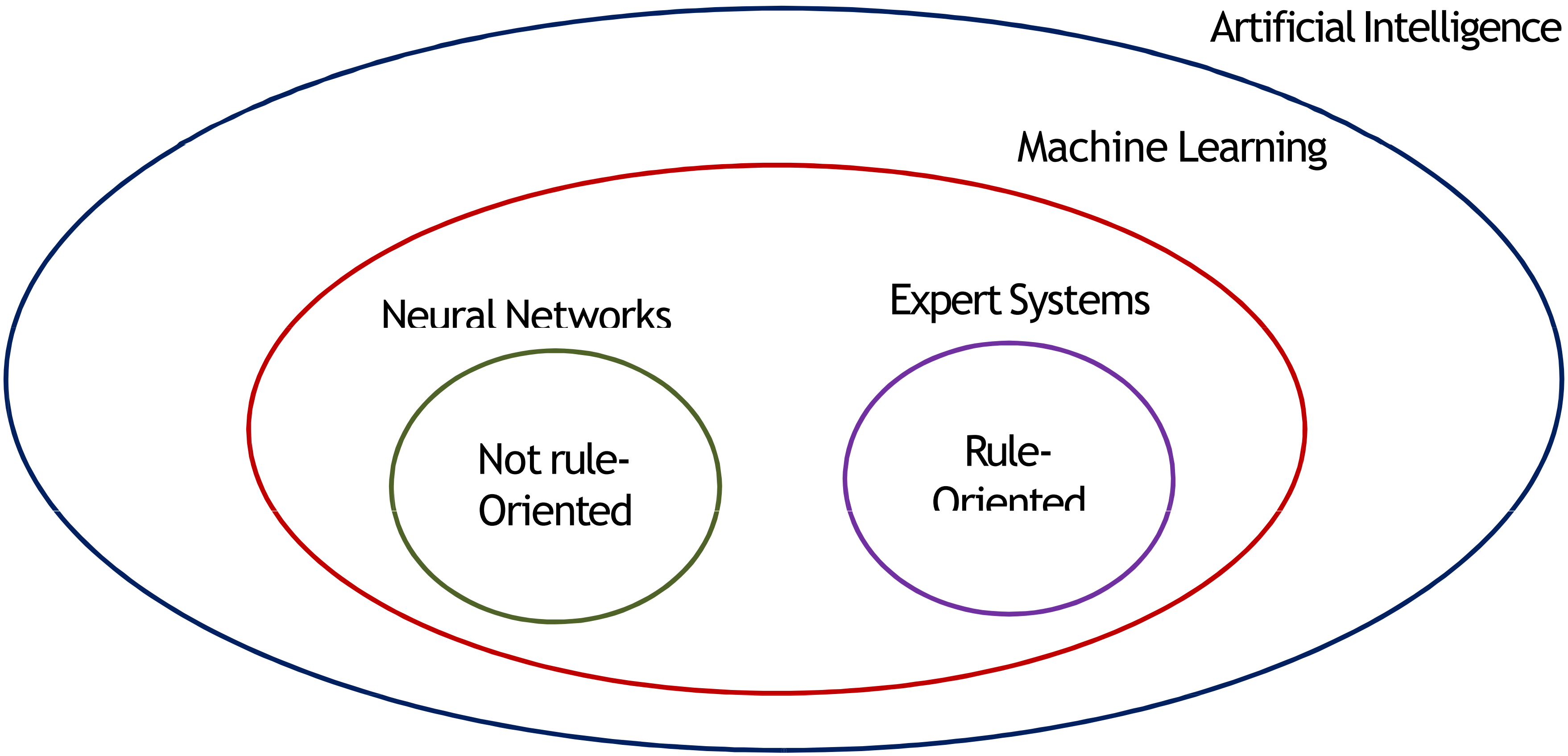
Neural Networks

Artificial Neural Networks (ANN)
Big Picture



What are ANNs

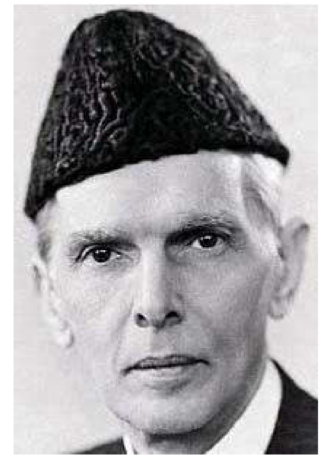
- *"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.*
  M. Caudill 1989
- Computational model based on the structure and functions of biological Neural Networks
- They are considered nonlinear statistical data modeling tools where the complex relationships between inputs and outputs are modeled or patterns are found

Biological Inspiration

Biological neural networks

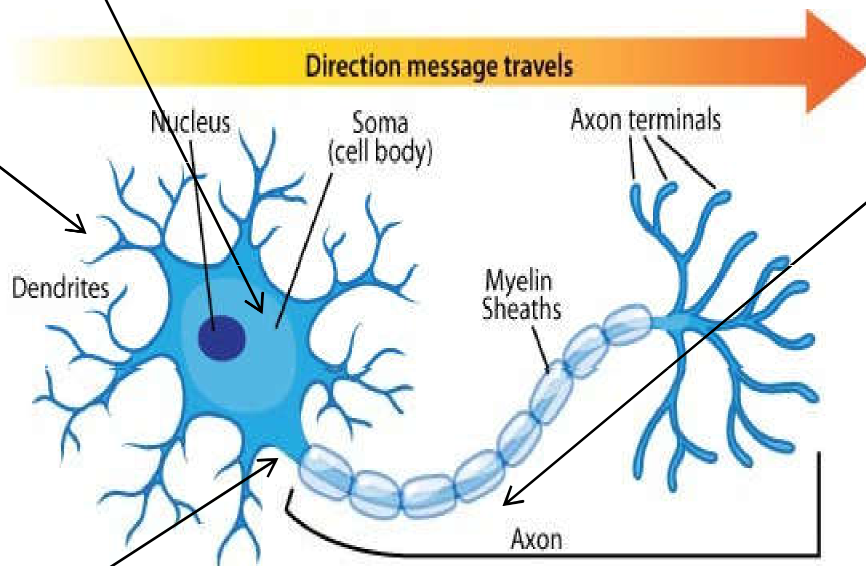
- About 10^{11} neurons in human brain
- About $10^{14\sim15}$ interconnections
- Pulse-transmission frequency million times slower than electronic circuits
- Face recognition
  - hundred million second by human

    Network of artificial neuron operation speed only a few million second



Biological Neuron

- **cell body:** computes a non-linear function of its inputs
- **dendrites:** nerve fibres carrying electrical signals to the cell



- **axon:** single long fibre that carries the electrical signal from the cell body to other neurons
- **synapse:** the point of contact between the axon of one cell and the dendrite of another, regulating a chemical connection whose strength affects the input to the cell.

Biological Neuron

- A variety of different neurons exist (motor neuron, on-center off-surround visual cells…), with different branching structures

- The connections of the network and the strengths of the individual synapses establish the function of the network.

Brief History of ANN

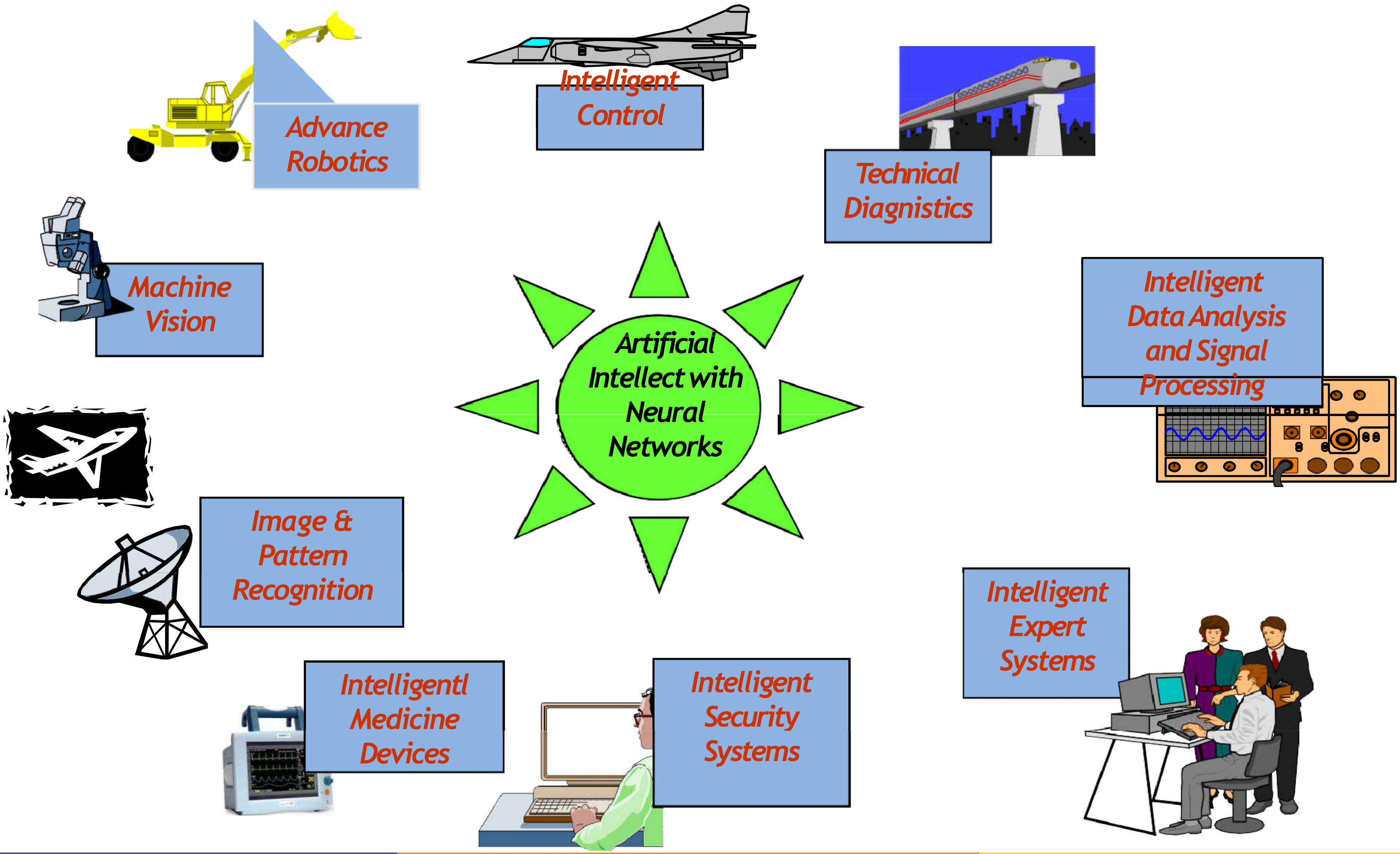
Brief History of ANN

- McCulloch and Pitts (1943) designed the first neural network
- Hebb (1949) who developed the first learning rule. If two neurons were active at the same time then the strength between them should be increased.
- Rosenblatt (1958) – introduced the concept of a perceptron which performed pattern recognition.
- Widrow and Hoff (1960) introduced the concept of the ADALINE (ADAptive Linear Element) . The training rule was based on the idea of Least-Mean-Squares learning rule which minimizing the error between the computed output and the desired output.
- Minsky and Papert (1969) stated that the perceptron was limited in its ability to recognize features that were separated by linear boundaries. “Neural Net Winter”
- Kohonen and Anderson – independently developed neural networks that acted like memories.
- Webros(1974) – developed the concept of back propagation of an error to train the weights of the neural network.
- McCelland and Rumelhart (1986) published the paper on back propagation algorithm. “Rebirth of neural networks”.
- Today - they are everywhere a decision can be made.

Source : G5AIAI - Introduction to Artificial Intelligence Graham Kendall:

Applications of ANN

Applications of Artificial Neural Networks



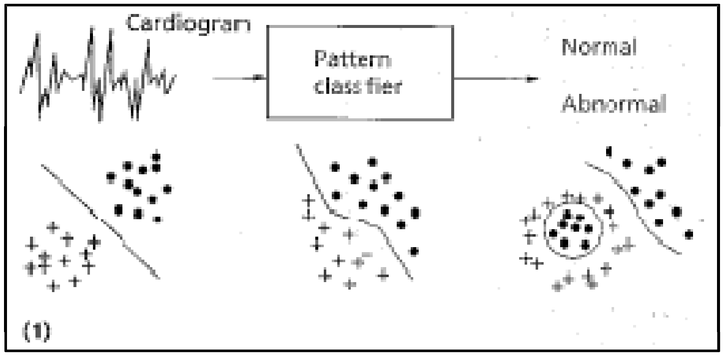
Applications of Artificial Neural Networks

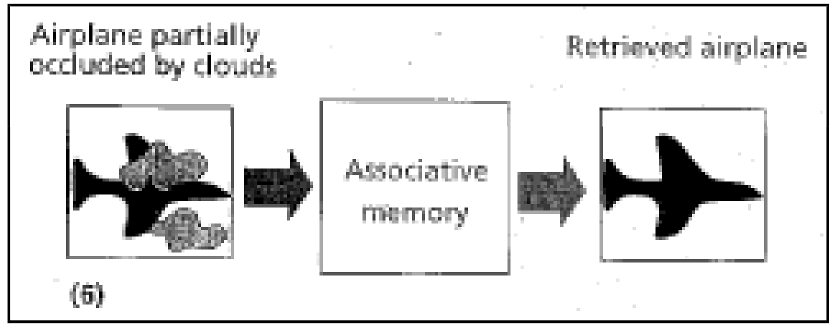
- Aerospace: aircraft autopilots, flight path simulations, aircraft control systems, autopilot enhancements, aircraft component simulations
- Banking: credit application evaluators
- Defense: guidance and control, **target detection and tracking**, object discrimination, sonar, radar and image signal processing including data compression, feature extraction and noise suppression, signal/image identification
- Financial: real estate appraisal, loan advisor, mortgage screening, stock market analysis, stock trading advisory systems
- Manufacturing: process control, process and machine diagnosis, visual quality inspection systems, computer chip quality analysis

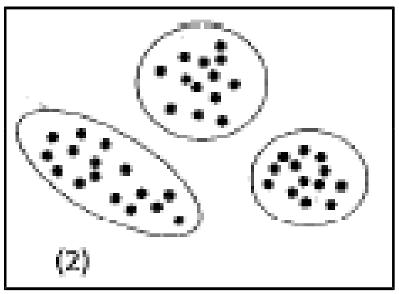
Applications of Artificial Neural Networks

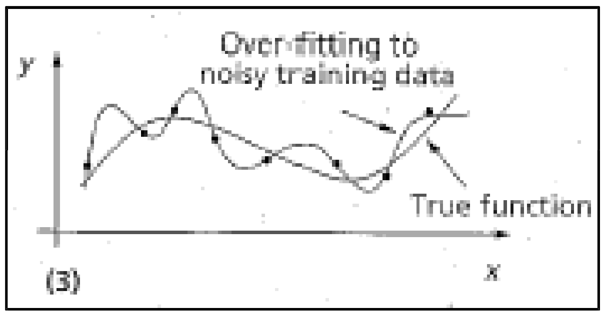
- Medical: cancer cell detection and analysis, EEG and ECG analysis, disease pathway analysis
- Communications: adaptive echo cancellation, image and data compression, speech synthesis, signal filtering
- Robotics: Trajectory control, manipulator controllers, vision systems
- Pattern Recognition: character recognition, speech recognition, voice recognition, facial recognition

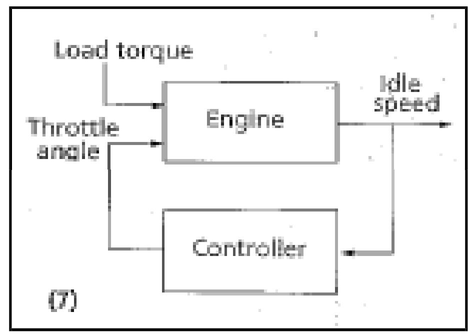
Challenging Problems

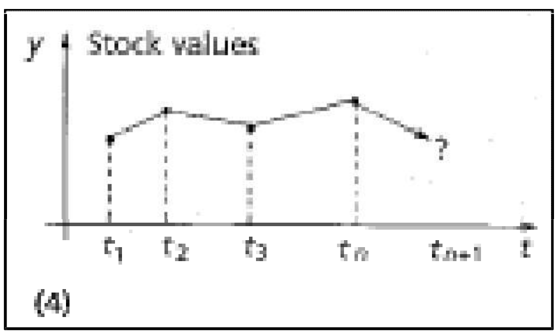


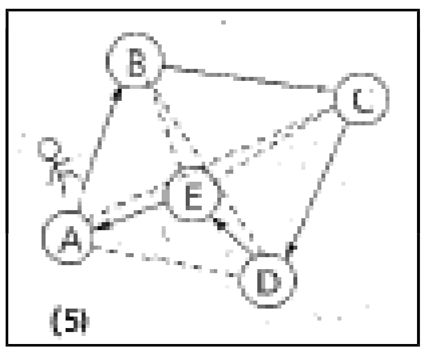












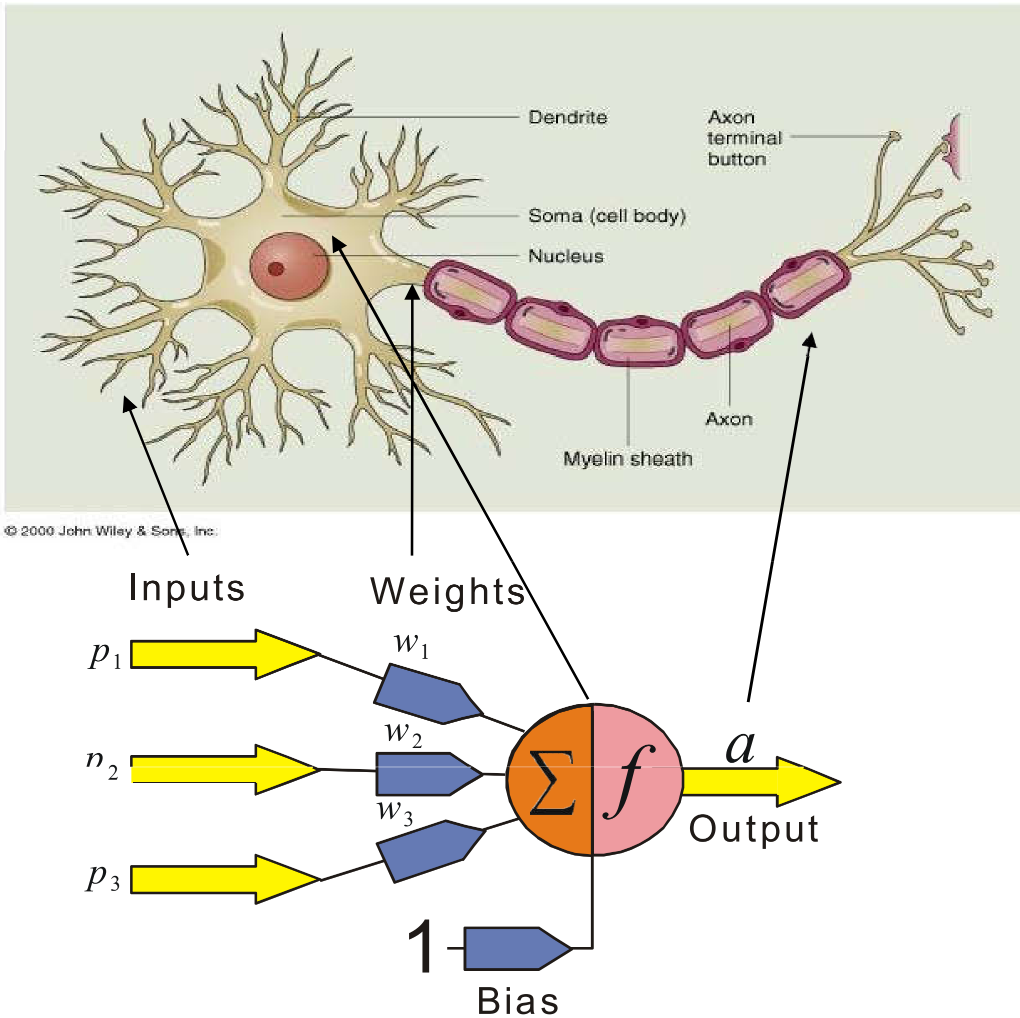
(1) Pattern classification
(2) Clustering/categorization
(3) Function approximation
(4) Prediction/forecasting
(5) Optimization (TSP problem)
(6) Retrieval by content
(7) Control

Neural Net vs. Von Neumann Computer

Neural Net	Von Neumann
Non-algorithmic	Algorithmic
Trained	Programmed with instructions
Memory and processing elements the same	Memory and processing separate
Pursue multiple hypotheses simultaneously	Pursue one hypothesis at a time
Fault tolerant	Non fault tolerant
Non-logical operation	Highly logical operation
Adaptation or learning	Algorithmic parameterization modification only
Seeks answer by finding minima in solution space	Seeks answer by following logical tree structure

Artificial Neuron

Artificial Neuron Model

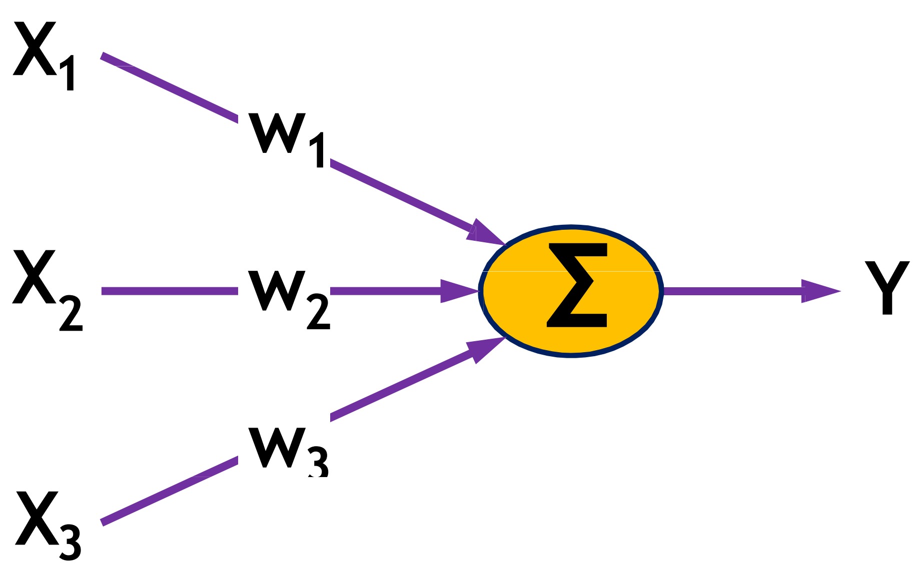


$$a = f(p_1 w_1 + p_2 w_2 + p_3 w_3 + b) = f\left(\sum p_i w_i + b\right)$$

Biological Neuron	Artificial Neuron
Dendrites	Input Connections
Soma	Activation and Activity Function
Axons	Output connections
Synaptic gap	Weights

Artificial Neuron

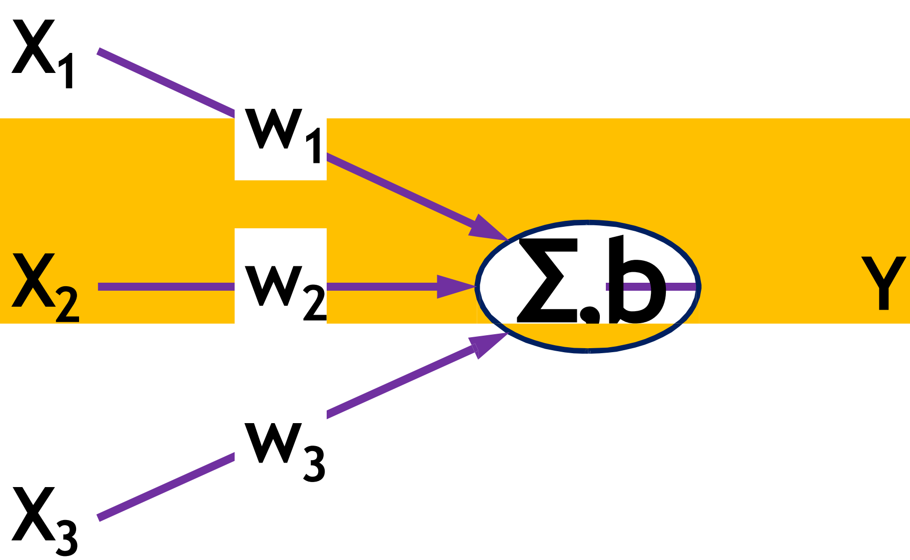
- Artificial Neuron is the basic information processing unit of the Neural Networks (NN). It is a non linear, parameterized function with restricted output range.



$$y = \sum_i x_i w_i$$

Adding bias

- Bias is like another weight. Its included by adding a component $x_0=1$ to the input vector X.

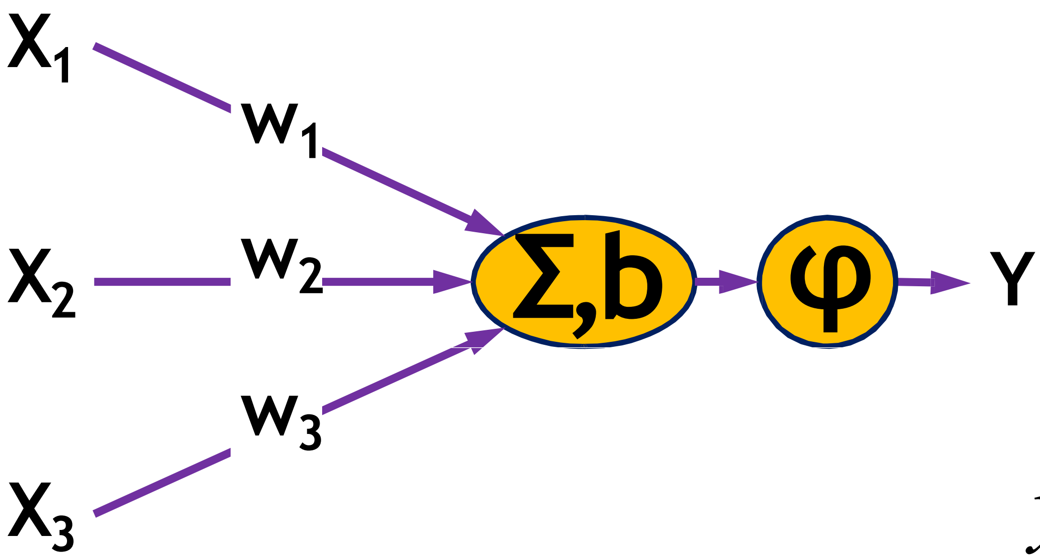


$$y=\sum_i x_i w_i - b$$

- Bias is of two types
  - Positive bias: increase the net input
  - Negative bias: decrease the net input

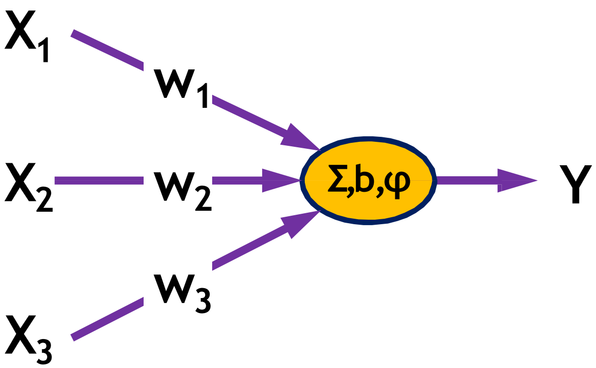
Adding an "activation" function

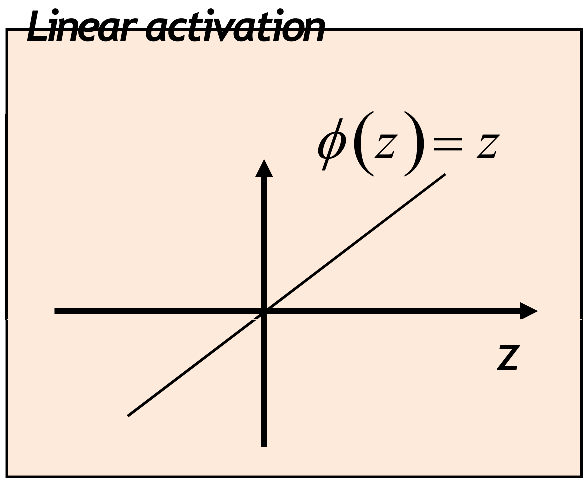
- Used to calculate the output response of a neuron.
- Sum of the weighted input signal is applied with an activation to obtain the response.
- Activation functions can be linear or non linear

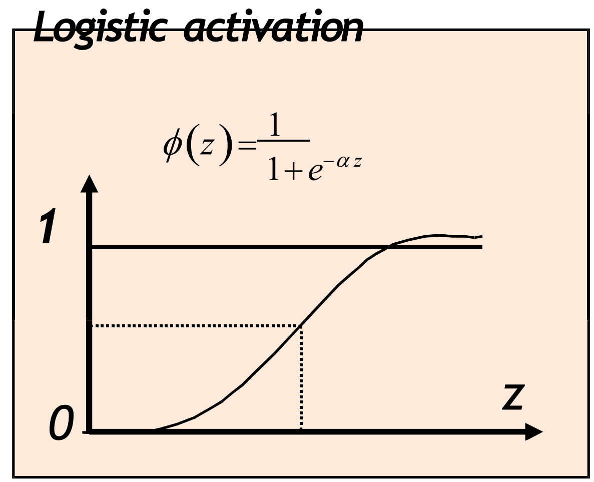


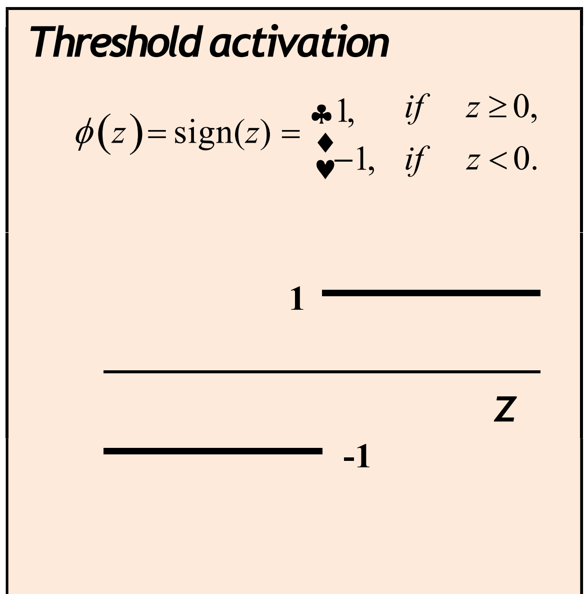
$$y = \Phi(\overbrace{\sum_i x_i w_i - b}^{Z,\ \text{(the field of the neuron)}})$$

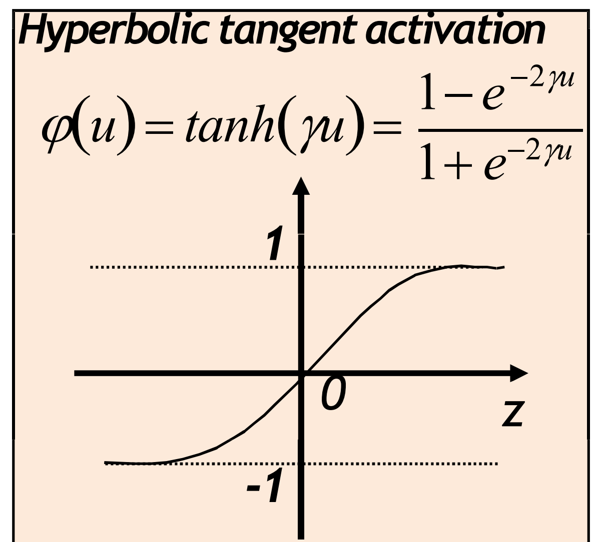
Common activation functions











Many types of activations functions are used:

linear: a = f(n) = n

Threshold: a = {1 if n >= 0 (hardlimiting)
0 if n < 0

sigmoid: a = 1/(1+e^{-n})

Artificial Neural Networks

Artificial Neural Networks

- **Artificial Neural Network (ANN):** is a machine learning approach that models human brain and consists of a number of artificial neurons that are linked together according to a specific network architecture.
- **Neuron** in ANNs tend to have fewer connections than biological neurons. each neuron in ANN receives a number of inputs.
- **An activation function** is applied to these inputs which results in activation level of neuron (output value of the neuron).
- Knowledge about the learning task is given in the form of examples called training examples.

Computing with Neural Units

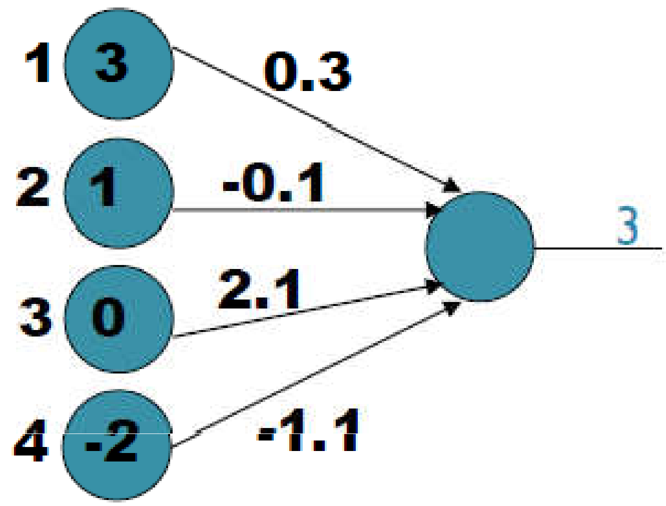
- Incoming signals to a unit are presented as inputs.
- How do we generate outputs?

- One idea: *Summed* *Weighted Inputs*.
- **Input**: (3, 1, 0, -2)
- **Processing**

3(0.3) + 1(-0.1) + 0(2.1) + -2(-1.1)

= 0.9 + (-0.1) + 0 + 2.2

- **Output**: 3



Activation Functions

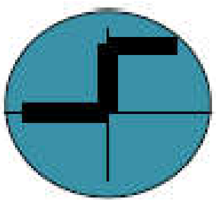
- Usually, do not just use weighted sum directly.
- Apply some function to the weighted sum before it is used (e.g., as output).
- ⓪ Call this the **activation function**.

$$f(x) = \begin{cases} 1 \text{ if } x \geq \theta \\ 0 \text{ if } x < \theta \end{cases}$$

θ Is called the threshold

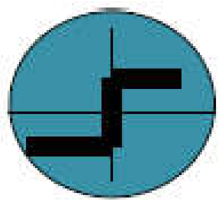
Step function

Activation Functions



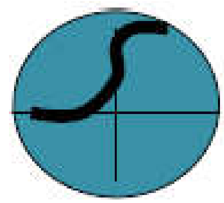
Step Function

step(x) =
1, if x >= threshold
0, if x < threshold
(in picture above, threshold = 0)



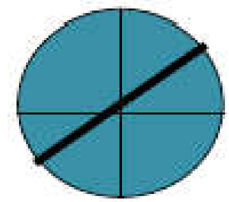
Sign Function

sign(x) =
+1, if x >= 0
-1, if x < 0



Sigmoid (Logistic) Function

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



Linear Function

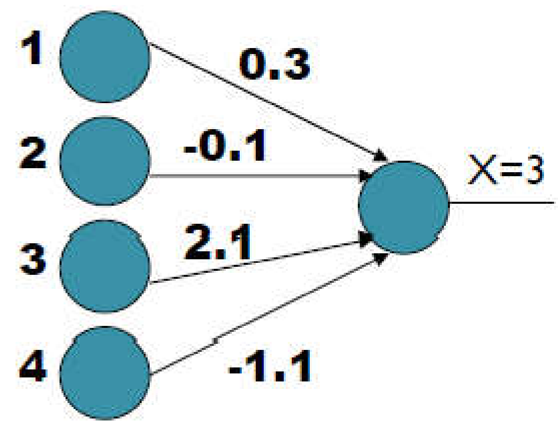
pl(x) = x

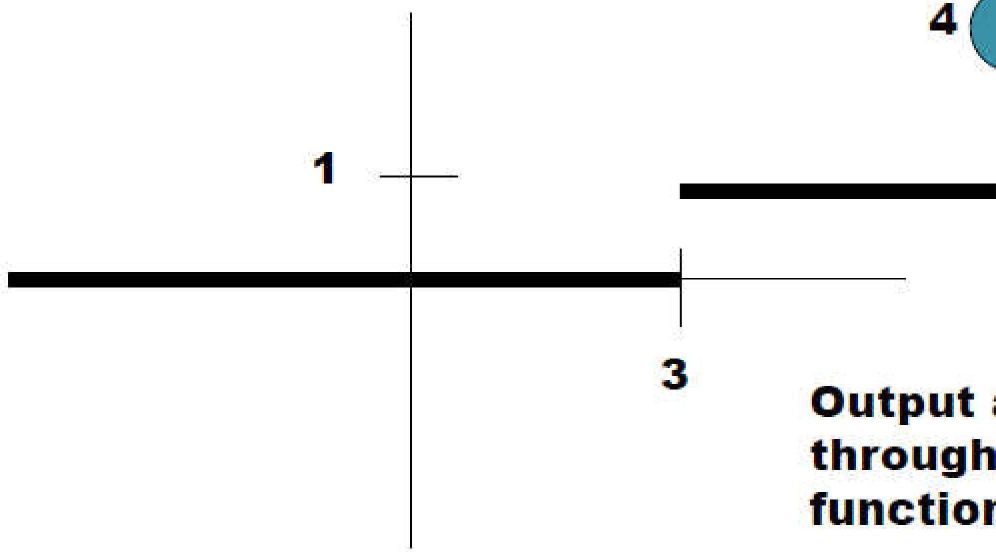
- The choice of activation function determines the ***Neuron Model*.**

Example (1): Step Function

- Let $\Theta = 3$

Input: (3, 1, 0, -2)





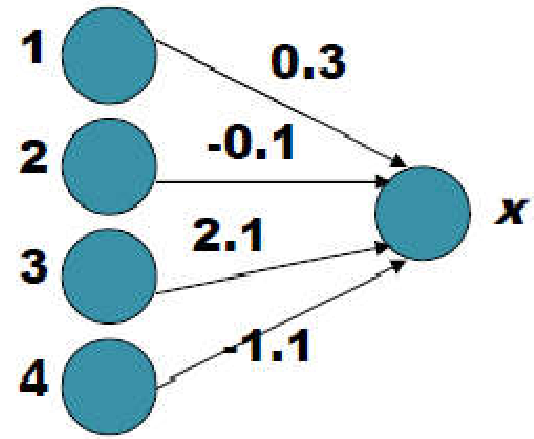
Output after passing through step activation function:

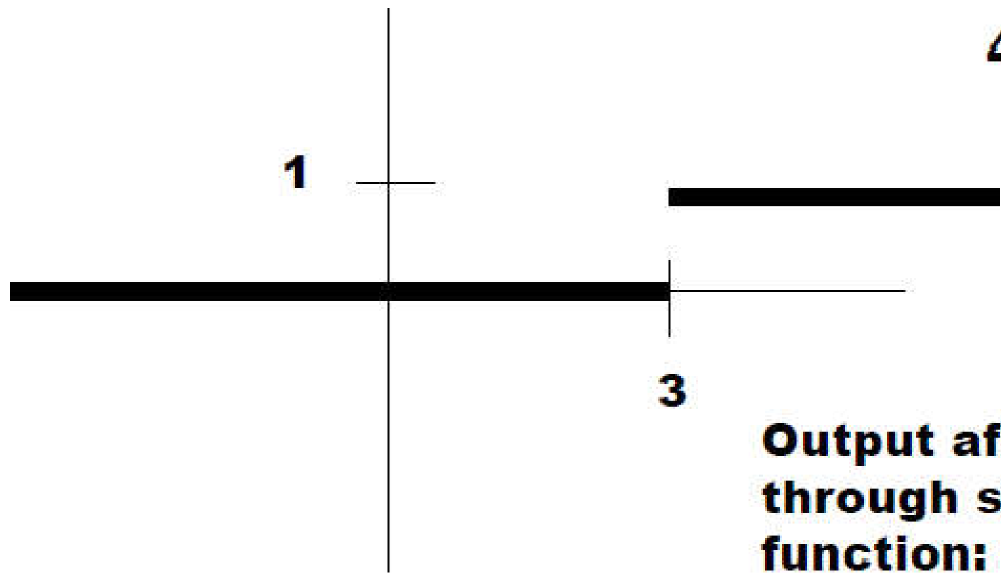
$$f(3) = 1$$

Example (2): Another Step Function

- Let $\Theta = 3$

Input: (0, 10, 0, 0)





Output after passing through step activation function:

$$f(x) = ?$$

Example (3): Sigmoid Function

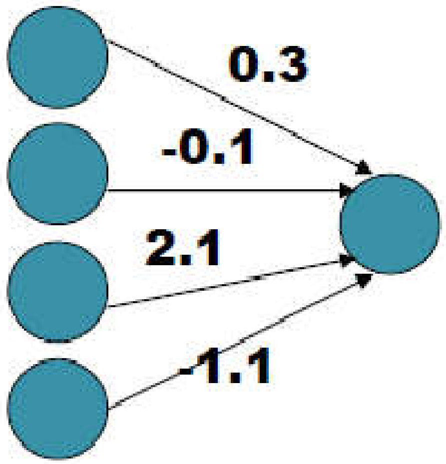
- The math of some neural nets requires that the activation function be continuously differentiable.

•A sigmoidal function often used to approximate the step function.

$$f(x) = \frac{1}{1 + e^{-\sigma x}}$$

σ Is the steepness parameter

Example (3): Sigmoid Function



$$\sigma = 2$$

***Input*: (3, 1, 0, -2)**

$$f(x) = \frac{1}{1 + e^{-2x}}$$

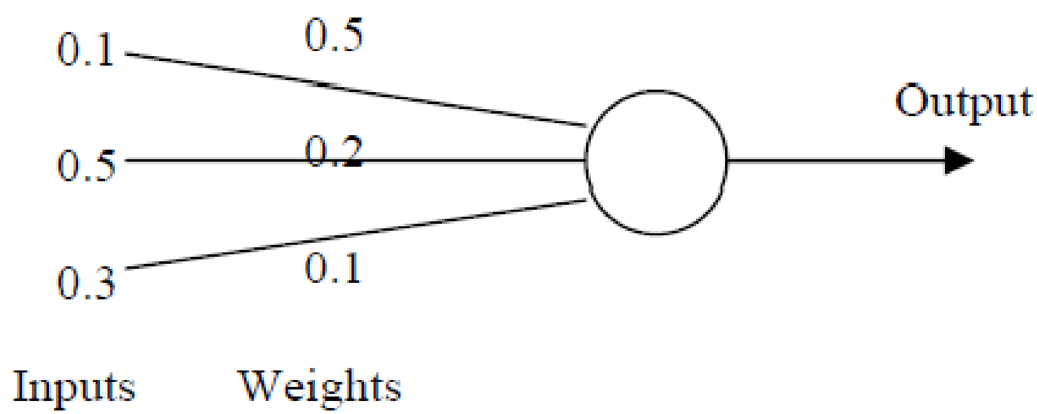
$$f(3) = \frac{1}{1 + e^{-2x}} = .998$$

***Input*: (0, 10, 0, 0)**

network output?

Example

- Calculate the output from the neuron below assuming a threshold of 0.5:



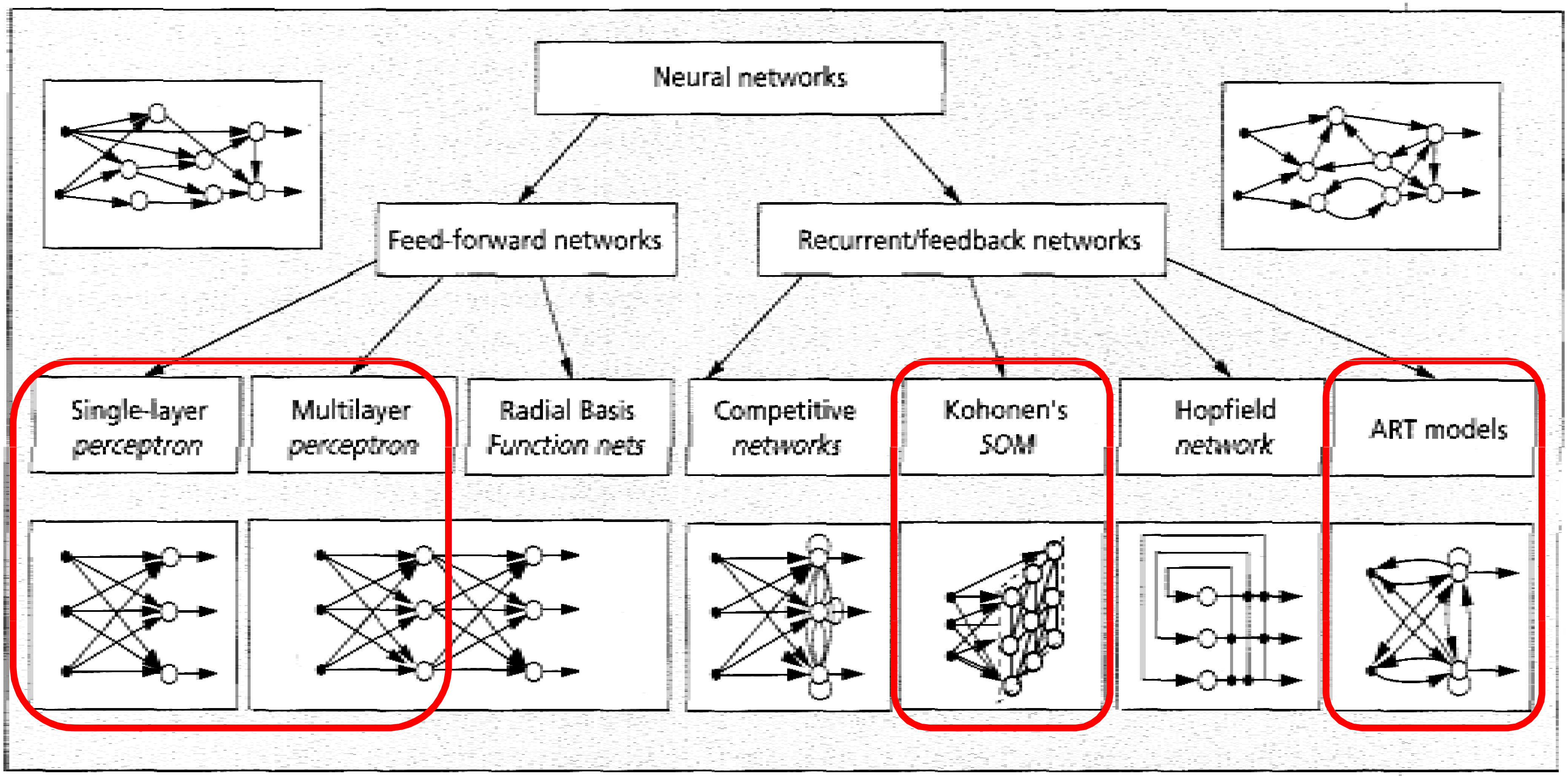
Sum = (0.1 x 0.5) + (0.5 x 0.2) + (0.3 x 0.1) = 0.05 + 0.1 + 0.03 = 0.18
Since 0.18 is less than the threshold, the Output = 0

Repeat the above calculation assuming that the neuron has a sigmoid output function:

Sum is still 0.18, but now $\text{Output} = \dfrac{1}{1+e^{-0.18}} = 0.545$

Network Architectures

Network architectures



A taxonomy of feed-forward and recurrent/feedback network architectures.

Network Architecture

- The Architecture of a neural network is linked with the learning algorithm used to train.

- **There are different classes of network architecture**:

  - Single-Layer Neural Networks.
  - Multi-Layer Neural Networks.

    - **The number of layers and neurons depend on the specific task.**

Single Layer Neural Network

- Single-layer feedforward network is **the simplest form** of a layered network.
- There are two layers:
  - Input Layer
  - Output Layer (Computation Nodes)
- It is ***feedforward***, means the information flow from input to output and not *viceversa*.
- Input layer of source nodes are **not counted** because no computation is performed.

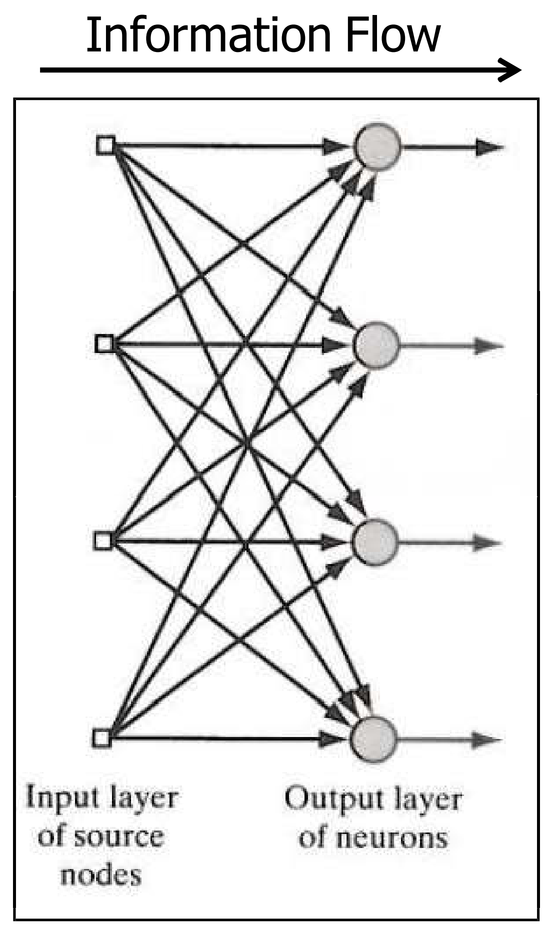


Figure 2.1 Feedforwad network with a single layer of neurons.

Multilayer Neural Network

- Multilayer feedforward networks has **one or more** hidden layers.
- Multilayer NN overcome the limitation of Single-Layer NN, they can handle non-linearly separable learning tasks.
- By adding hidden layers, the network is enabled to **extract higher-order statistics** from its input.
- In this structure, the **computation nodes** are called *hidden neurons* or *hidden units*.
- The example architecture in **Figure 2.2** is referred to as a 10-4-2 network:
  - 10 source nodes
  - 4 hidden neurons
  - 2 output neurons
- **Fully** Connected VS **Partially** Connected

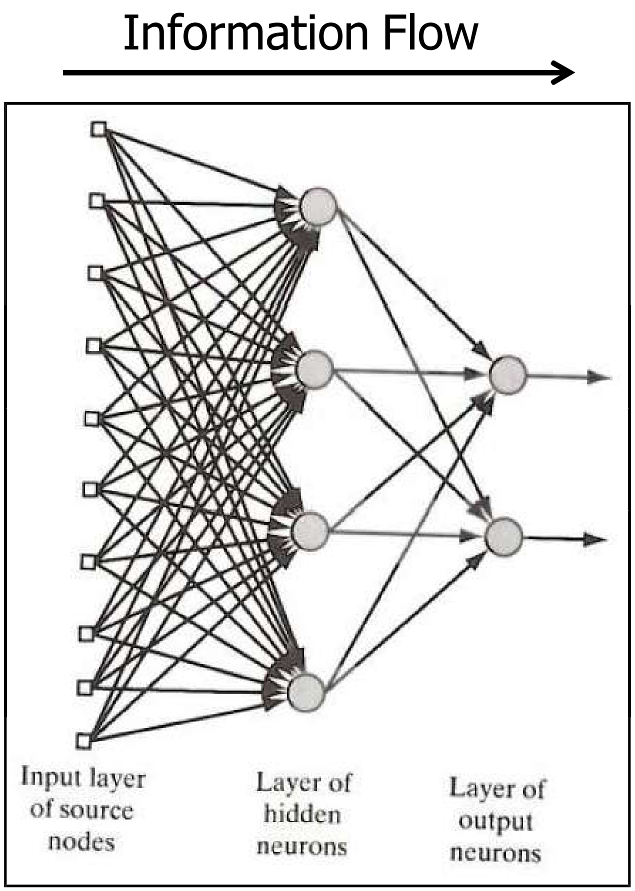


Figure 2.2 Fully connected feedforward network with one hidden layer and one output layer.

Recurrent Network

- Recurrent neural network is **different** from *feedforward* neural network because it has at least one *feedback* loop.
- The presence of feedback loop has a **profound impact** on the **learning capability** of the network and its performance.
- The feedback loops involve the use of particular branches composed of **unit-time delay elements** (denoted by z^{-1})
- Structure at Figure 2.3:
  - No *self-feedback* loops in the network
  - No *hidden* neurons

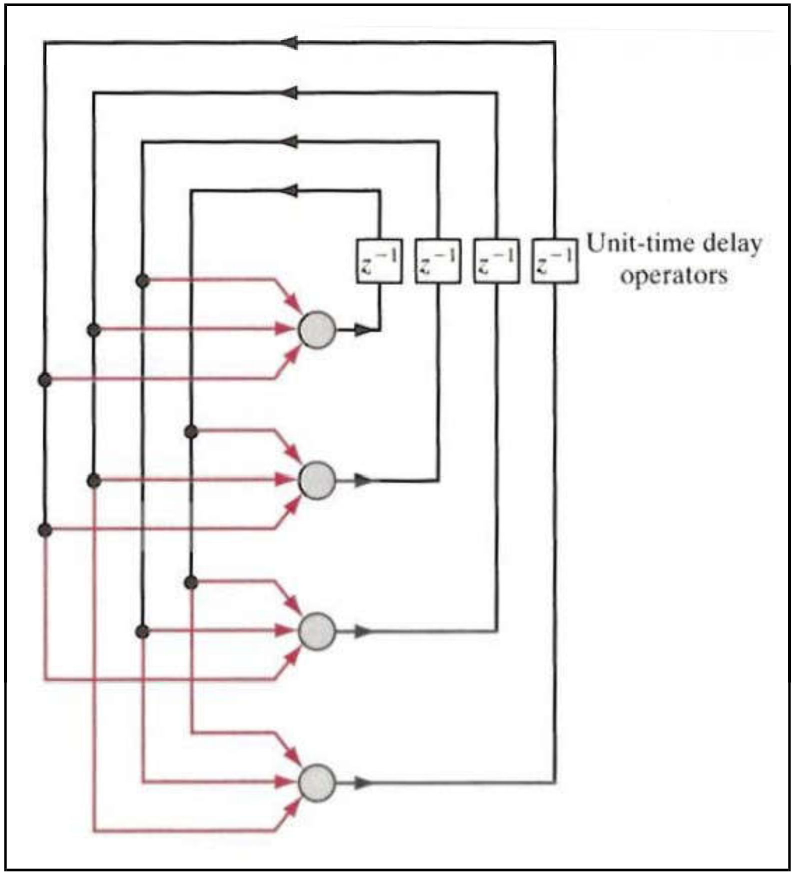


Figure 2.3 Recurrent network with no self-feedback loops and no hidden neurons.

Recurrent Network

- Structure at Figure 2.4:
  - Contains *self-feedback* loops in the network
  - Contains *hidden* neurons

- The feedback connections **originate** from the hidden neurons as well as from the output neurons.

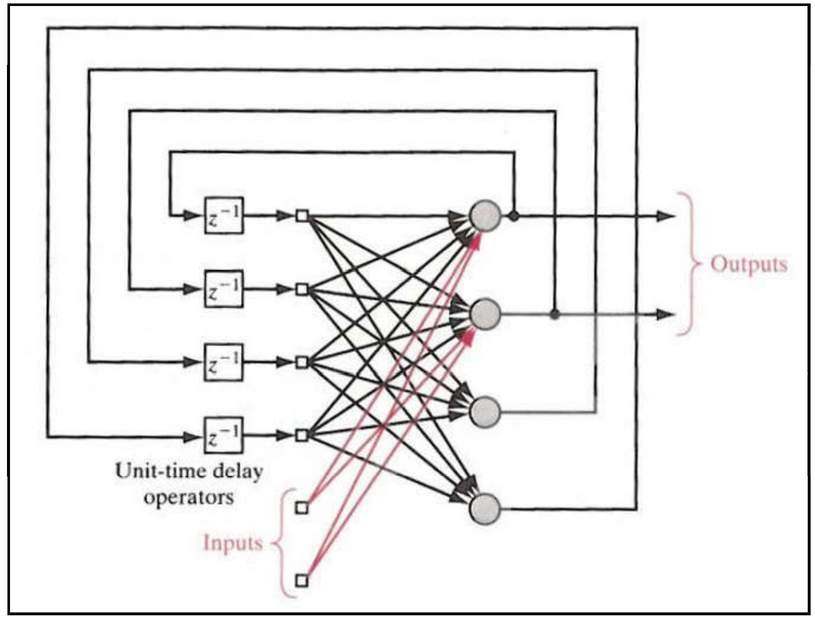
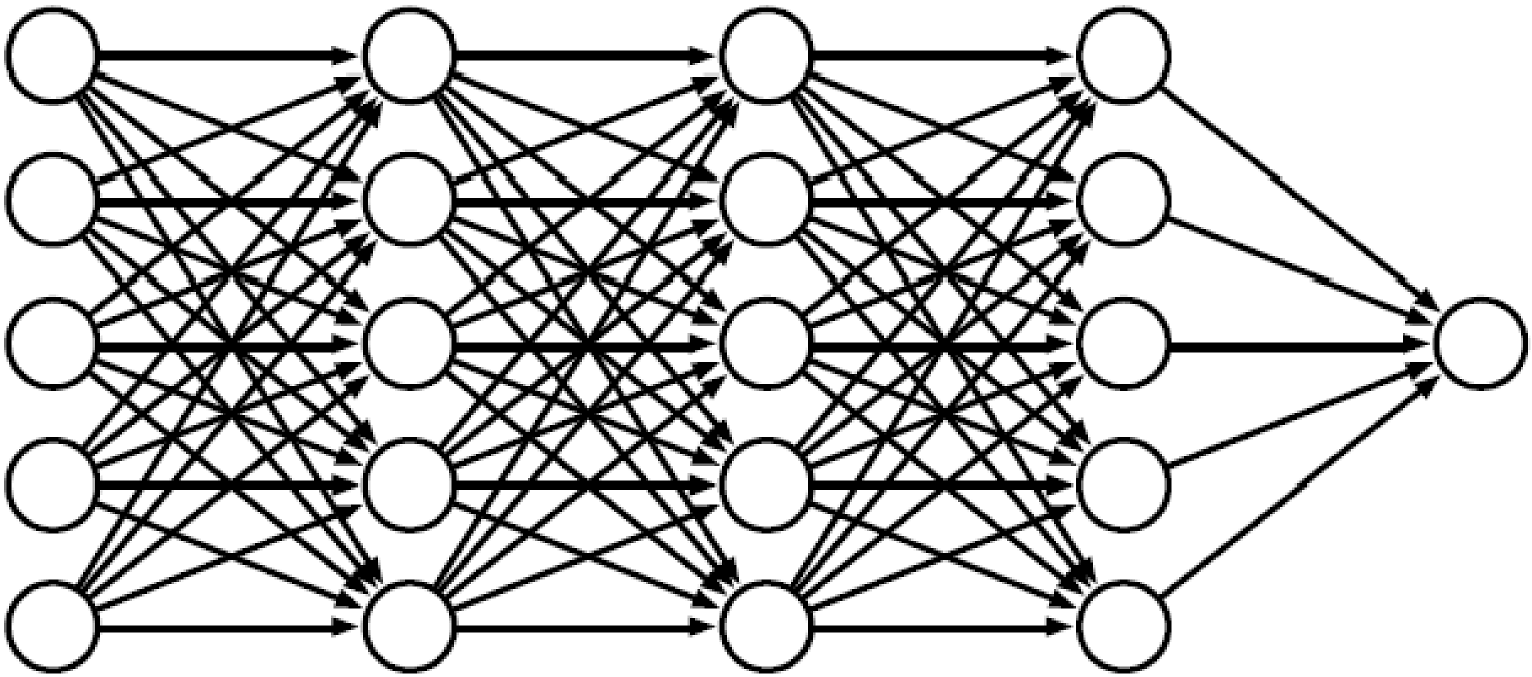


Figure 2.4 Recurrent network with hidden neurons.

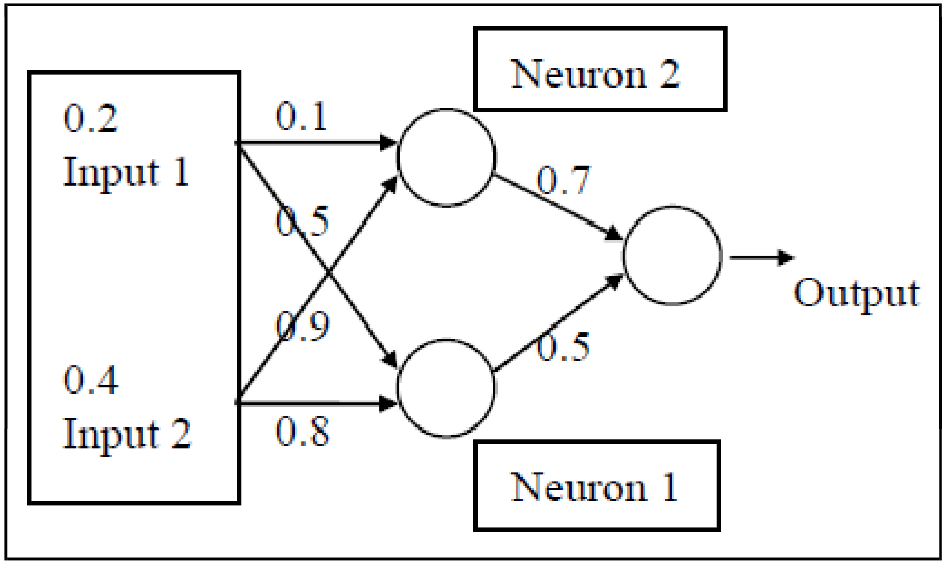
Deep Learning

More layers =deep learning



Example of multilayer ANN

- Calculate the output from this network assuming a Sigmoid Squashing Function.



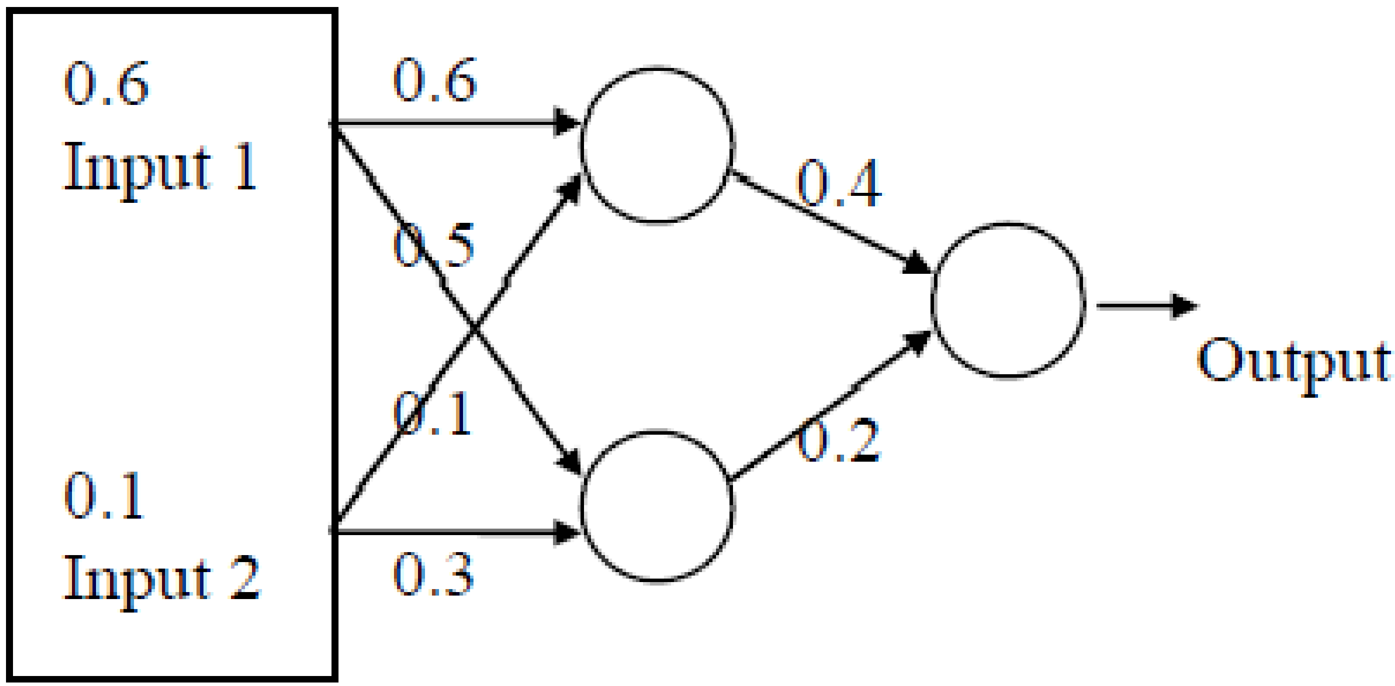
Input to neuron 1 = (0.2 x 0.5) + (0.4 x 0.8) = 0.42. Output = $\frac{1}{1+e^{-0.42}} = 0.603$

Input to neuron 2 = (0.2 x 0.1) + (0.4 x 0.9) = 0.38. Output = $\frac{1}{1+e^{-0.38}} = 0.594$

Input to final neuron = (0.594 x 0.7) + (0.603 x 0.5) = 0.717.

Final Output = $\frac{1}{1+e^{-0.717}} = 0.672$

Exercise of multilayer ANN



- Try calculating the output of this network yourself.

Backpropagation

An efficient method of implementing gradient descent for neural networks

$$w_{i\to j} = w_{i\to j} - r\delta_j y_i$$ **Descent rule**

$$\delta_j = \frac{ds(z_j)}{dz_j}\sum_k \delta_k w_{j\to k}$$ **Backprop rule**

$$(i) \xrightarrow[y_i]{w_{i\to j}} (j) \xrightarrow[y_i]{w_{j\to k}} (k)$$

j k

y_i is x_i for input layer

1. Initialize weights to small random values
2. Choose a random sample input feature vector
3. Compute total input (z_j) and output (y_j) for each unit (forward prop)
4. Compute δ_n for output layer $\delta_n = \frac{ds(z_n)}{dz_n}(y_n - y_n^*) = y_n(1-y_n)(y_n - y_n^*)$
5. Compute δ_j for preceding layer by backprop rule (repeat for all layers)
6. Compute weight change by descent rule (repeat for all weights)

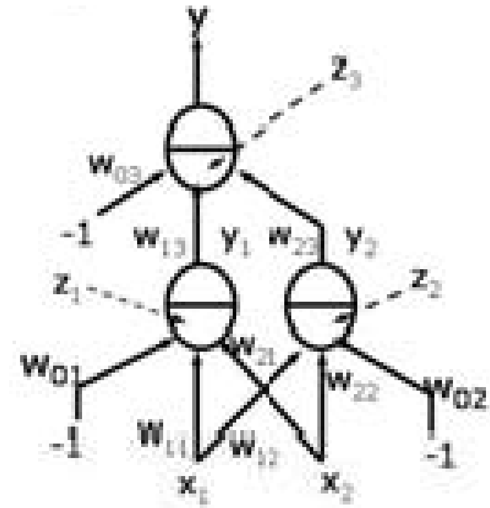
Backpropagation

First do forward propagation:
Compute z_i and y_i given x_k, w_{ij}

$$\delta_3 = y(1-y)(y-y^m)$$

$$\delta_2 = y_2(1-y_2)\delta_3 w_{23}$$

$$\delta_1 = y_1(1-y_1)\delta_3 w_{13}$$



$$w_{03} = w_{03} - r\delta_3(-1) \qquad w_{13} = w_{13} - \eta\delta_3 y_1 \qquad w_{23} = w_{23} - \eta\delta_3 y_2$$

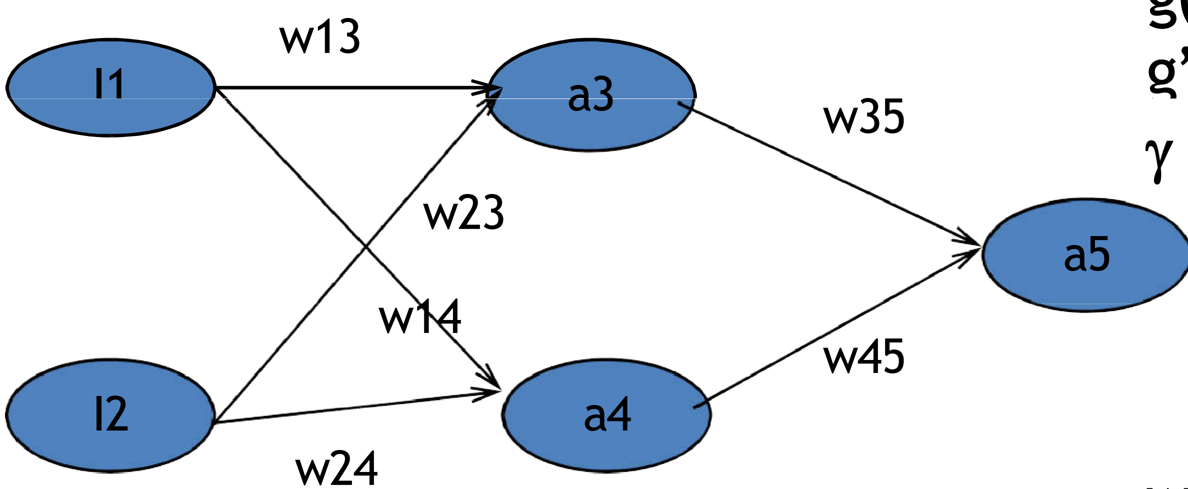
$$w_{02} = w_{02} - r\delta_2(-1) \qquad w_{12} = w_{12} - \eta\delta_2 x_1 \qquad w_{22} = w_{22} - \eta\delta_2 x_2$$

$$w_{01} = w_{01} - r\delta_1(-1) \qquad w_{11} = w_{11} - \eta\delta_1 x_1 \qquad w_{21} = w_{21} - \eta\delta_1 x_2$$

Compare to the direct derivation earlier

Note that all computations are local!

Back Propagation Formula Example



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

$g'(x) = (1-x)*x$

γ is the learning rate

$a4=g(z4)=g(x1*w14+x2*w24)$

$a3=g(z3)=g(x1*w13+x2*w23)$

$a5=g(z5)=g(a3*w35+a4*w45)$

$\otimes 5=error*g'(z5)=error*a5*(1-a5)$

$\otimes 4= \otimes 5*w45*g'(z4)=\otimes 5*w45*a4*(1-a4)$

$\otimes 3=\otimes 5*w35*a3*(1-a3)$

$w35= w35 +\gamma*a3*\otimes 5$

$w45= w45 +\gamma*a4*\otimes 5$

$w13= w13 + \gamma*x1*\otimes 3$

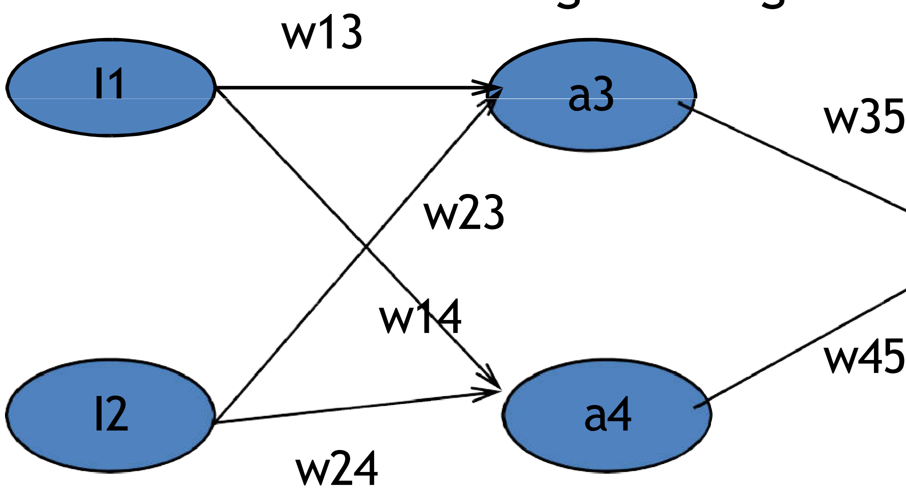
$w23= w23 + \gamma*x2*\otimes 3$

$w14= w14 + \gamma*x1*\otimes 4$

$w24= w24 +\gamma*x2*\otimes 4$

Example BP

Example: all weights are 0.1 except w45=1; γ=0.2
Training Example: (x1=1,x2=1;a5=1)
g is the sigmoid function



a3=g(z3)=g(x1*w13+x2*w23)=g(0.2)=0.550
a4=g(z4)=g(x1*w14+x2*w24)=g(0.2)=0.550
a5=g(z5)=g(a3*w35+a4*w45)=g(0.605)=0.647
⊗5=*error**g'(z5)
=*error**a5*(1-a5)
=0.353*0.647*0.353=0.08
⊗3=⊗5*w35*a3*(1-a3)=0.002
⊗4=⊗5*w45*a4*(1-a4)=0.02

a5 is 0.6483 with the adjusted weights!

w35= w35 +γ*a3*⊗5
=0.1+0.2*0.55*0. 08=0.109
w45= w45 +γ*a4*⊗5=1.009

w13= w13 + γ*x1*⊗3=0.1004
w23= w23 + γ*x2*⊗3=0.1004
w14= w14 +γ*x1*⊗4=0.104
w24= w24 +γ*x2*⊗4=0.104

a3'=g(0.2044)=0.551
a4'=g(0.2044)=0.551
a5'=g(0.611554)=0.6483

ANN Capabilities & Limitations

Main capabilities of ANN includes:

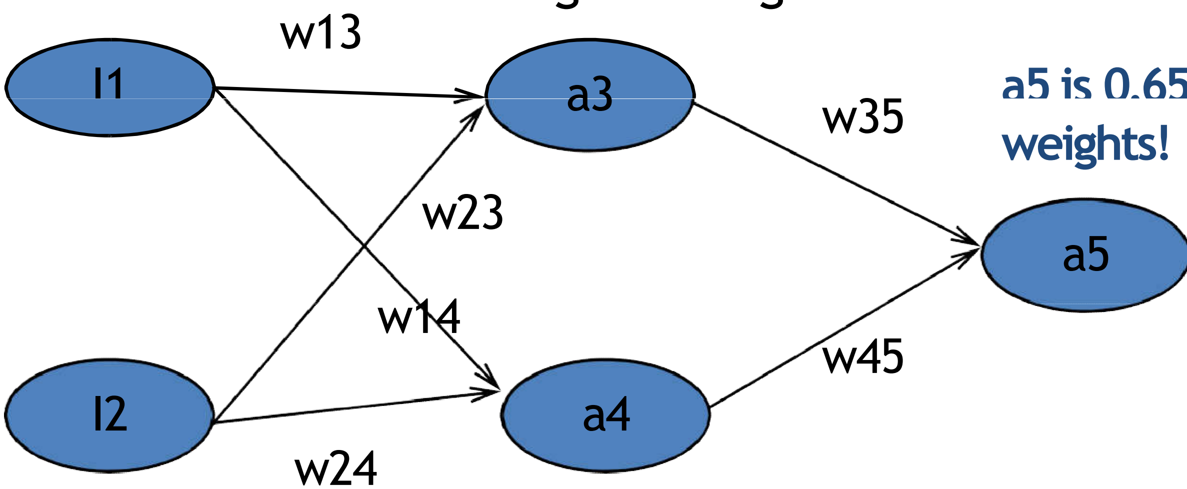
- Learning.
- Generalization capability.
- Noise filtering.
- Parallel processing.
- Distributed knowledge base.
- Fault tolerance.

Main problems includes:

- Learning sometimes difficult/slow.
- Limited storage capability.
- Do not do well at tasks that are not done well by people
- Lack explanation capabilities
- Limitations and expense of hardware technology restrict most applications to software simulations
- Training time can be excessive and tediou
- Usually requires large amounts of training and test data

Example BP

Example: all weights are 0.1 except w45=1; γ=1
Training Example: (x1=1,x2=1;a5=1)
g is the sigmoid function



a5 is 0.6594 with the adjusted weights!

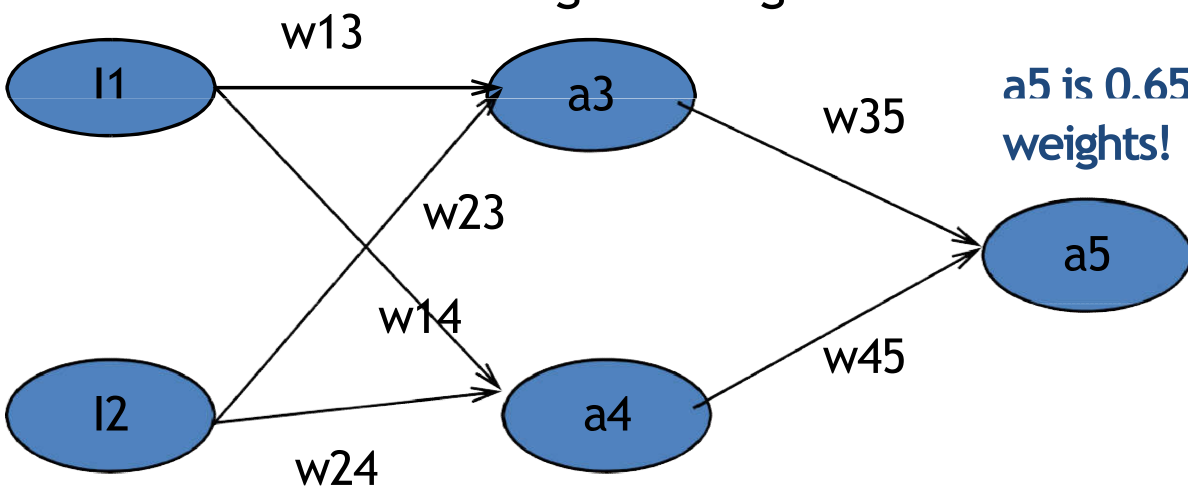
a3=g(z3)=g(x1*w13+x2*w23)=g(1*0.1+1*0.1)=g(0.2)=0.550

a4=g(z4)=g(x1*w14+x2*w24)=g(1*0.1+1*0.1)=g(0.2)=0.550

a5=g(z5)=g(a3*w35+a4*w45)=g(0.555*0.1+0.555*1)=g(0.605)=0.647

Example BP

Example: all weights are 0.1 except w45=1; γ=1
Training Example: (x1=1,x2=1;a5=1)
g is the sigmoid function



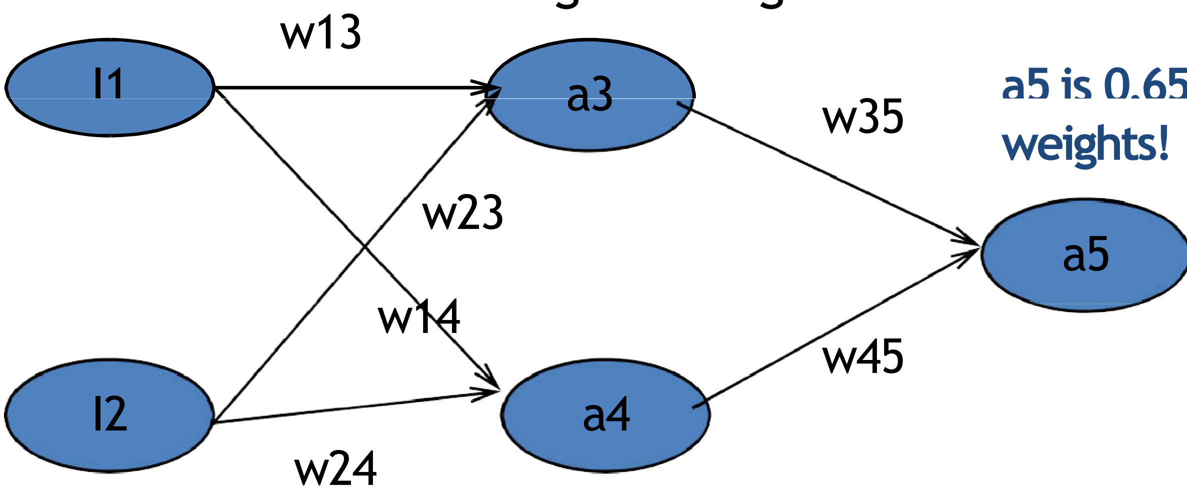
a5 is 0.6594 with the adjusted weights!

a3=g(z3)=0.550
a4=g(z4)=0.550
a5=g(z5)=0.647

⊗5=*error**g'(z5)
=*error**a5*(1-a5)
= 0.353*0.647*0.353=0.08
⊗3=⊗5*w35*a3*(1-a3)=0.00
2⊗4=⊗5*w45*a4*(1-a4)=0.0
2

Example BP

Example: all weights are 0.1 except w45=1; γ=1
Training Example: (x1=1,x2=1;a5=1)
g is the sigmoid function



a5 is 0.6594 with the adjusted weights!

a3=g(z3)=0.550
a4=g(z4)=0.550
a5=g(z5)=0.647

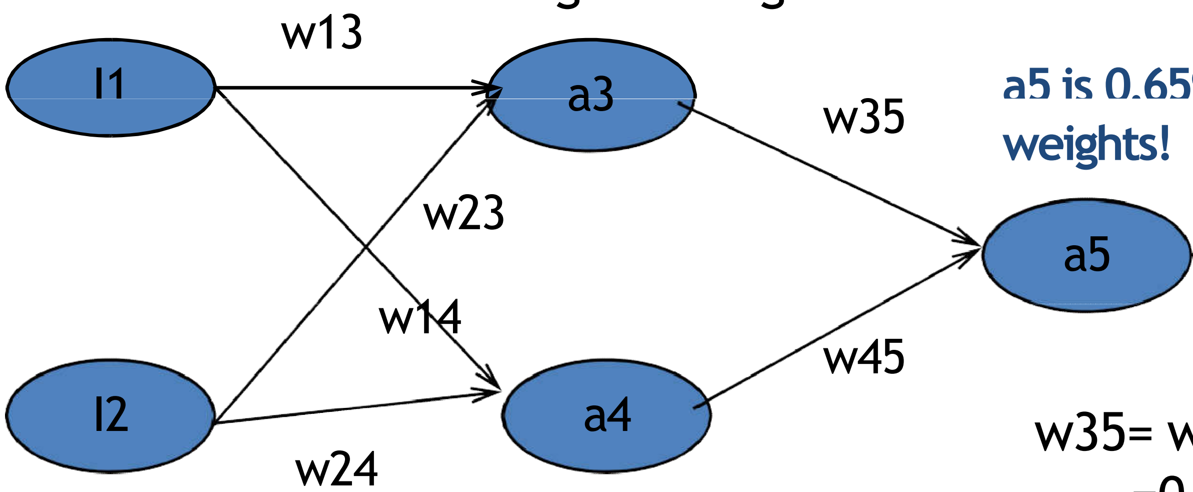
⊗5=*error**g'(z5)
=*error**a5*(1-a5)
= 0.353*0.647*0.353=0.08
⊗3=⊗5*w35*a3*(1-a3)=0.002
⊗4=⊗5*w45*a4*(1-a4)=0.02

w35= w35 +γ*a3*⊗5
=0.1+1*0.55*0.08=0.145
w45= w45 +γ*a4*⊗5=1.045

w13= w13 + γ*x1*⊗3=0.102
w23= w23 + γ*x2*⊗3=0.102
w14= w14 +γ*x1*⊗4=0.12
w24= w24 +γ*x2*⊗4=0.12

Example BP

Example: all weights are 0.1 except w45=1; γ=1
Training Example: (x1=1,x2=1;a5=1)
g is the sigmoid function



a5 is 0.6594 with the adjusted weights!

w35= w35 +γ*a3*⊗5
=0.1+1*0.55*0.08=0.145
w45= w45 +γ*a4*⊗5=1.045

a3=g(z3)=g(x1*w13+x2*w23)=g(0.2)=0.555

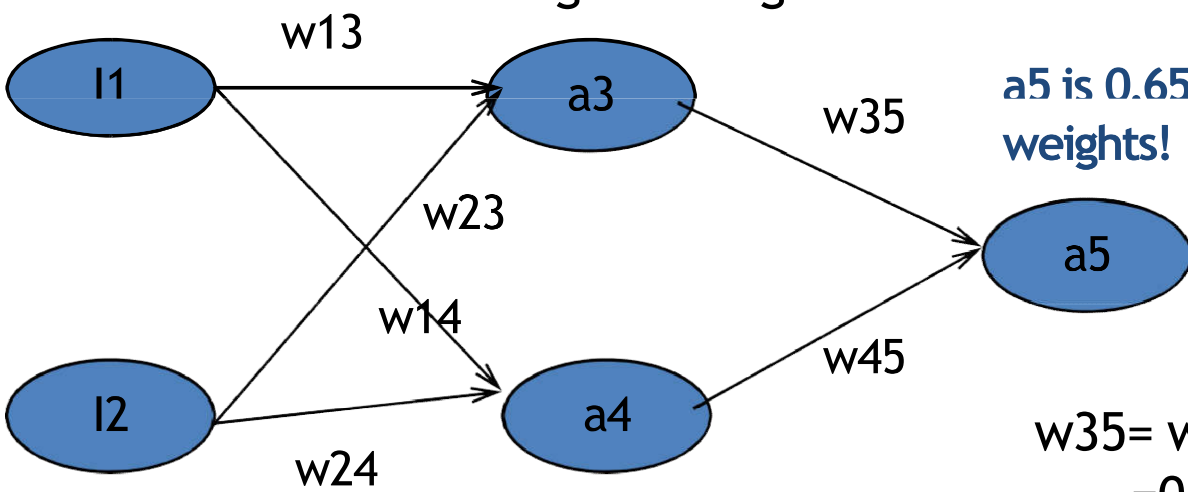
a4=g(z4)=g(x1*w14+x2*w24)=g(0.2)=0.555

a5=g(z5)=g(a3*w35+a4*w45)=g(0.605)=0.6594

w13= w13 +γ*x1*⊗3=0.102
w23= w23 +γ*x2*⊗3=0.102
w14= w14 +γ*x1*⊗4=0.12
w24= w24 +γ*x2*⊗4=0.12

Example BP

Example: all weights are 0.1 except w45=1; γ=1
Training Example: (x1=1,x2=1;a5=1)
g is the sigmoid function



a5 is 0.6594 with the adjusted weights!

$$w35 = w35 + \gamma * a3 * \otimes 5$$
$$= 0.1 + 1 * 0.55 * 0.08 = 0.145$$

$$w45 = w45 + \gamma * a4 * \otimes 5 = 1.045$$

$$w13 = w13 + \gamma * x1 * \otimes 3 = 0.102$$
$$w23 = w23 + \gamma * x2 * \otimes 3 = 0.102$$
$$w14 = w14 + \gamma * x1 * \otimes 4 = 0.12$$
$$w24 = w24 + \gamma * x2 * \otimes 4 = 0.12$$

$$a3 = g(z3) = g(x1 * w13 + x2 * w23)$$
$$= g(1 * 0.102 + 1 * 0.12) = g(0.222) = 0.555$$
$$a4 = g(z4) = g(x1 * w14 + x2 * w24)$$
$$= g(1 * 0.102 + 1 * 0.12) = g(0.222) = 0.555$$
$$a5 = g(z5) = g(a3 * w35 + a4 * w45)$$
$$= g(0.555 * 0.145 + 0.555 * 1.045)$$
$$= g(0.66045) = 0.6594$$

Hence, current error =(1-0.6594)=0.3406 which less than 0.353