“

Industrial Process Control MDP 454

If you have a smart project, you can say "I'm an engineer" ”

Lecture 10

Staff boarder

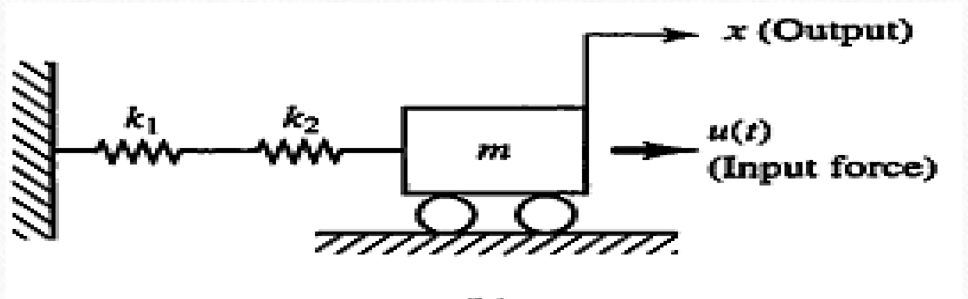
Dr. Mostafa Elsayed Abdelmonem

Industrial Process Control
MDP 454

- **Lecture aims:**
  - Solve simple problems using the Artificial Intelligent.
  - Formulate advanced problems for neural networks.

Quiz

- Obtain state-space representations of the mechanical systems shown in Figure



(b)

Biological Inspiration

Idea : To make the computer more robust, intelligent, and learn, …

Let’s model our computer software (and/or hardware) after the brain

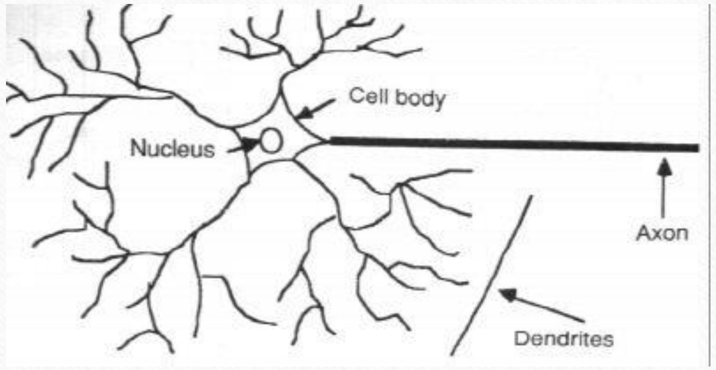
“My brain: It's my second favorite organ.”

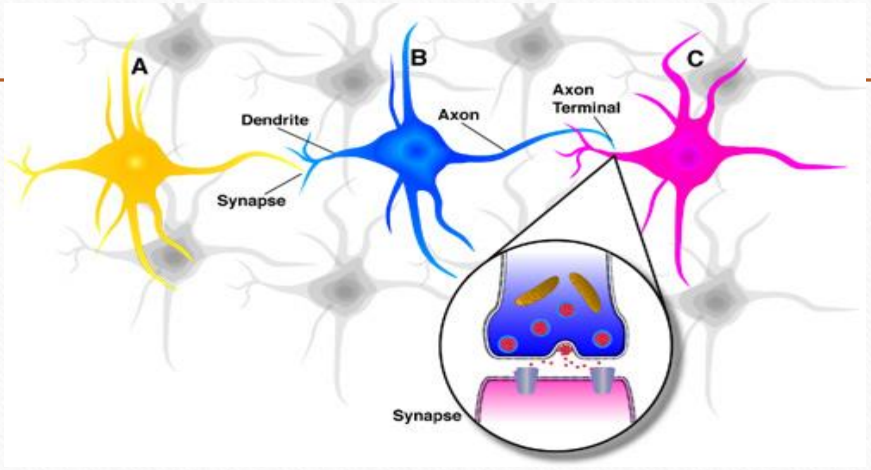
- Woody Allen, from the movie Sleeper

Comparison of Brains and Traditional Computers

200 billion neurons, 32 trillion synapses	1 billion bytes RAM but trillions of bytes on disk
Element size: 10^{-6} m	Element size: 10^{-9} m
Energy use: 25W	Energy watt: 30-90W (CPU)
Processing speed: 100 Hz	Processing speed: 10^{9} Hz
Parallel, Distributed	Serial, Centralized
Fault Tolerant	Generally not Fault Tolerant
Learns: Yes	Learns: Some
Intelligent/Conscious: Usually	Intelligent/Conscious: Generally No

How the Human Brain learns





- In the human brain, a typical neuron collects signals from others through a host of fine structures called *dendrites*.
- The neuron sends out spikes of electrical activity through a long, thin stand known as an *axon*, which splits into thousands of branches.
- At the end of each branch, a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity in the connected neurons.

History

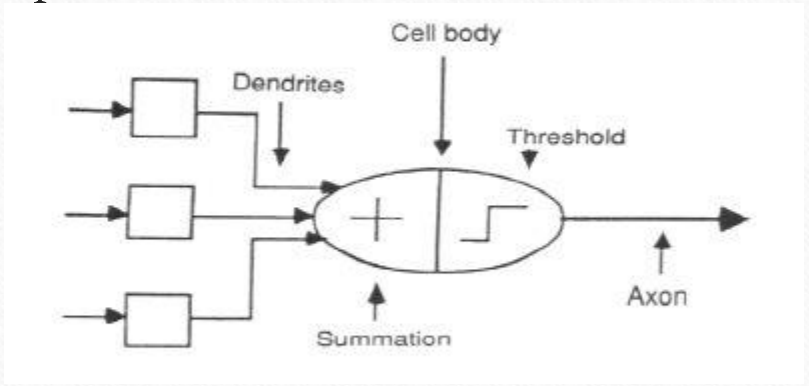
- 1943: McCulloch–Pitts "neuron"
  - Started the field
- 1962: Rosenblatt's perceptron
  - Learned its own weight values; convergence proof
- 1969: Minsky & Papert book on perceptrons
  - Proved limitations of single-layer perceptron networks
- 1982: Hopfield and convergence in symmetric networks
  - Introduced energy-function concept
- 1986: Backpropagation of errors
  - Method for training multilayer networks
- Present: Probabilistic interpretations, Bayesian and spiking networks

Properties of Artificial Neural Networks

- High level abstraction of neural input-output transformation
  - Inputs $\rightarrow$ weighted sum of inputs $\rightarrow$ nonlinear function $\rightarrow$ output
    - Typically no spikes
    - Typically use implausible constraints or learning rules
- Often used where data or functions are uncertain
    - Goal is to learn from a set of training data
    - And to generalize from learned instances to new unseen data
- Key attributes
    - Parallel computation
    - Distributed representation and storage of data
    - Learning (networks adapt themselves to solve a problem)
    - Fault tolerance (insensitive to component failures)

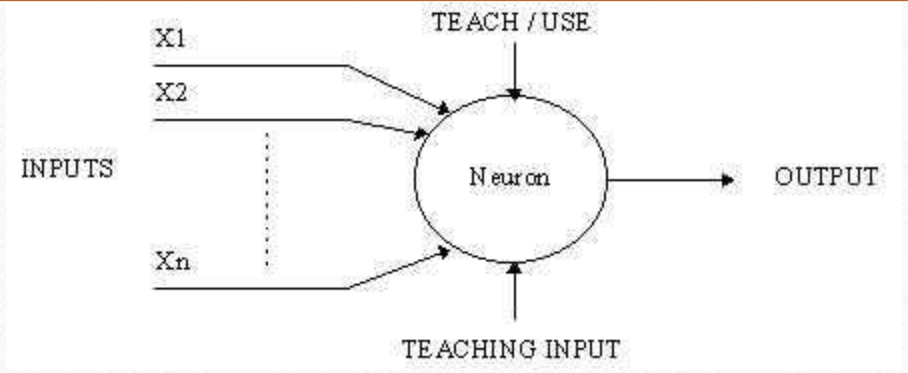
A Neuron Model

- When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.



- We conduct these neural networks by first trying to deduce the essential features of neurons and their interconnections.
- We then typically program a computer to simulate these features.

A Simple Neuron

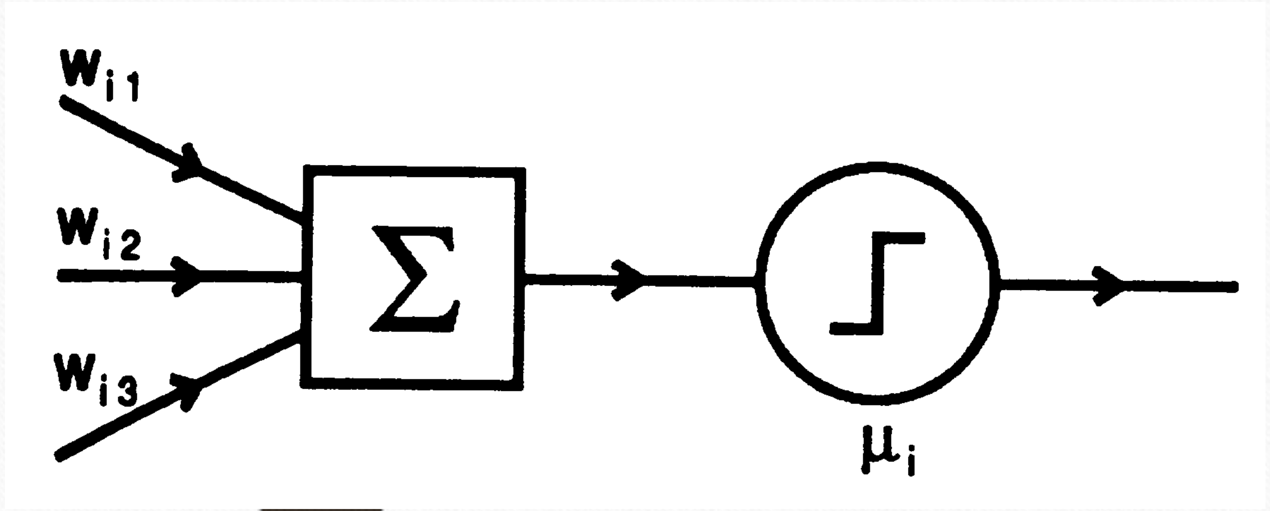


- An artificial neuron is a device with many inputs and one output.
- The neuron has two modes of operation;
    - the training mode and
    - the using mode.

A Simple Neuron

McCulloch–Pitts “neuron” (1943)

- Attributes of neuron
  - m binary inputs and 1 output (0 or 1)
  - Synaptic weights w_{ij}
  - Threshold μ_i



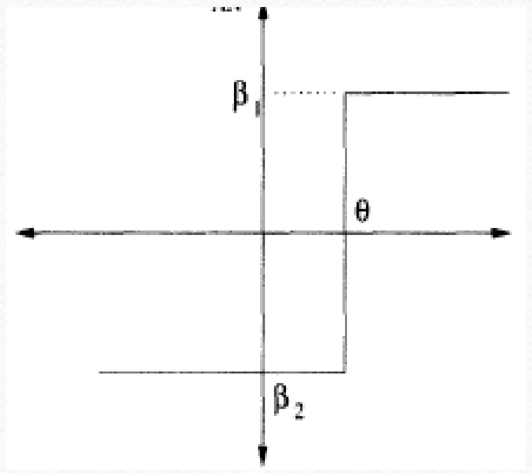
ARTIFICIAL NEURAL NETWORKS

Activation Functions

Step Function

$y = f(act) \quad = \beta_1 \quad$ If $act \geq 0$

$\qquad\qquad\quad = \beta_2 \quad$ If $act < 0$



For the step function only one of the two scalar values are possible at the output

Usually (β_1, β_2) are taken as (1, -1) or (1, 0)

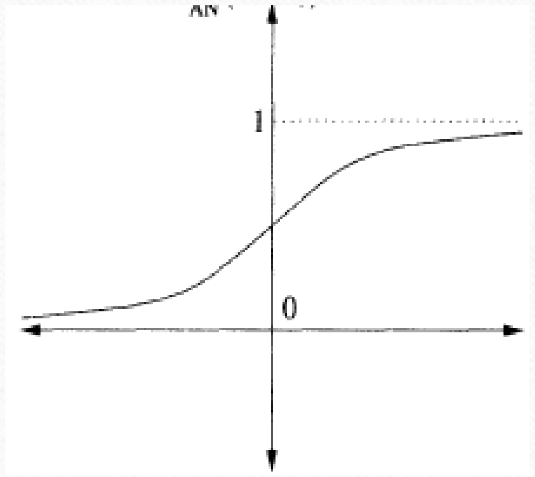
ARTIFICIAL NEURAL NETWORKS

Activation Functions

Sigmoid Function (Logistic function)

$$y = f(act) = \frac{1}{1 + e^{-\lambda(act)}}$$

The sigmoid function is a continuous version of the ramp function



The parameter λ controls the steepness of the function. Large λ makes it almost a unit step function.
Usually $\lambda = 1$

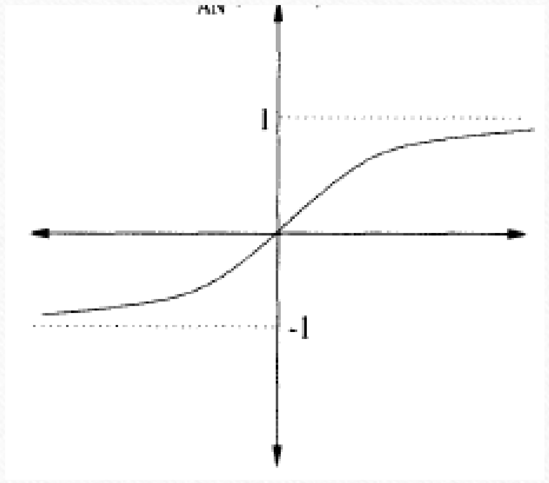
ARTIFICIAL NEURAL NETWORKS

Activation Functions

Hyperbolic Tangent Function

$$y = f(act) = \frac{e^{\lambda(act)} - e^{-\lambda(act)}}{e^{\lambda(act)} + e^{-\lambda(act)}}$$

$$= \left(\frac{2}{1 + e^{-\lambda(act)}} \right) - 1$$



The output of this function is in the range (-1, 1)

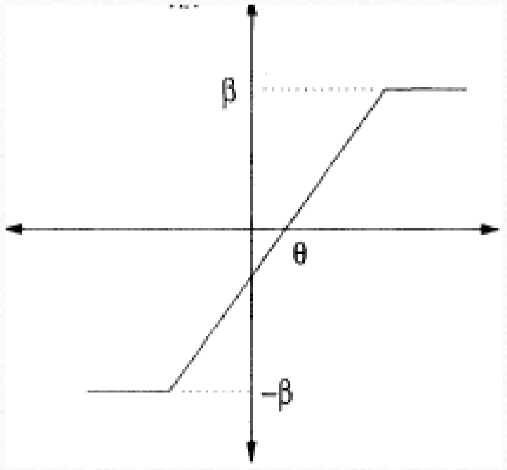
ARTIFICIAL NEURAL NETWORKS

Activation Functions

Ramp Function mxzc

$$y = f(act) = \begin{cases} \beta & \text{If } act \geq \beta \\ act & \text{If } -\beta < act < \beta \\ -\beta & \text{If } act \leq -\beta \end{cases}$$

It is a combination of the linear and step functions



ARTIFICIAL NEURAL NETWORKS

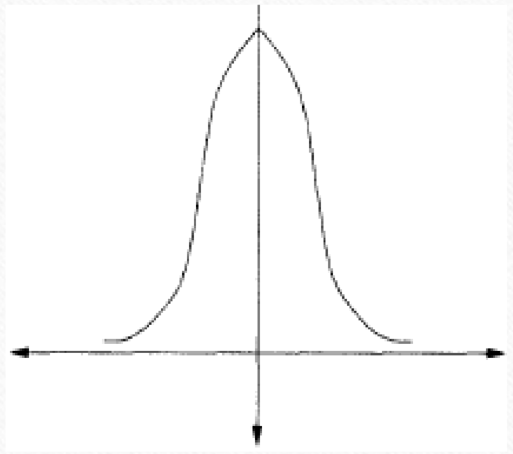
Activation Functions

Gaussian Function

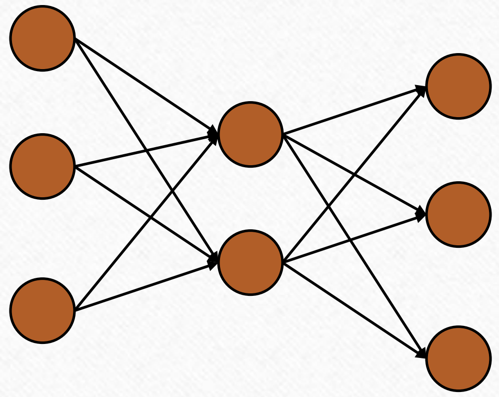
$y = f(act) = \quad e^{-\theta}$

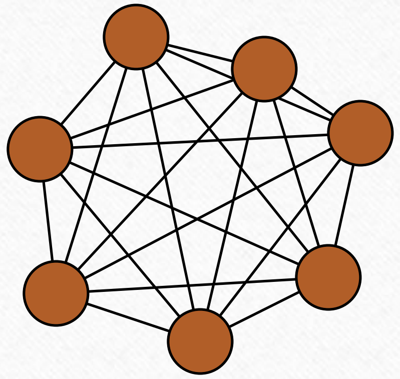
where $\theta = (act)^2/\sigma^2$

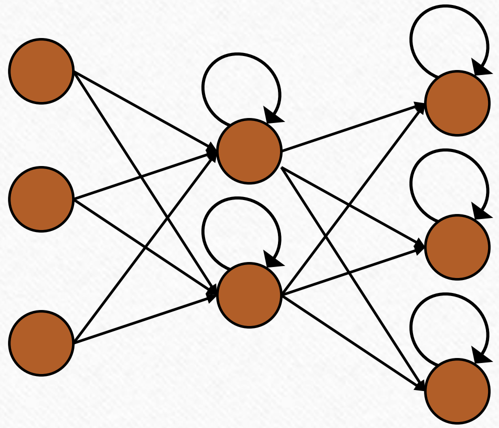
Where σ^2 is the variance of the Gaussian distribution

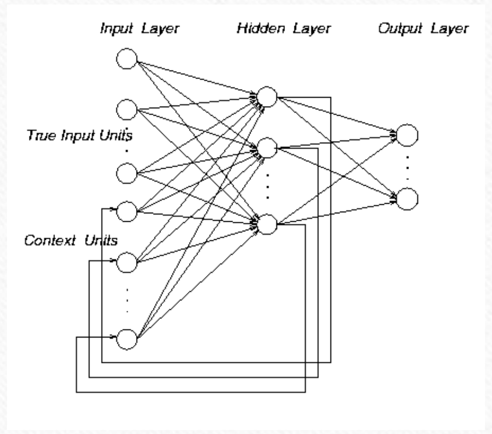


Topologies of Neural Networks









feedforward
(directed, a-cyclic)

completely
connected

recurrent
(feedback connections)

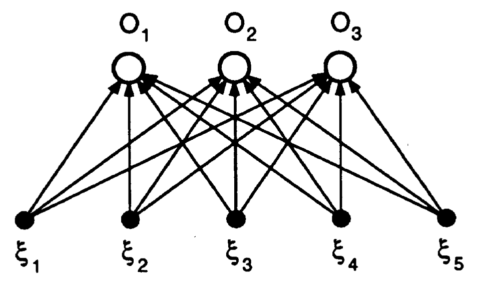
Networks Types

- Feedforward versus recurrent networks
  - Feedforward: No loops, input → hidden layers → output
  - Recurrent: Use feedback (positive or negative)
- Continuous versus spiking
  - Continuous networks model mean spike rate (firing rate)
    - Assume spikes are integrated over time
  - Consistent with rate-code model of neural coding
- Supervised versus unsupervised learning
  - Supervised networks use a "teacher"
    - The desired output for each input is provided by user
  - Unsupervised networks find hidden statistical patterns in input data
    - Clustering, principal component analysis

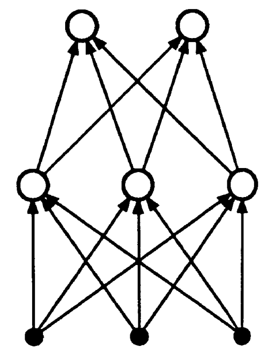
Perceptrons

- Attributes
  - Layered feedforward networks
  - Supervised learning
    - Hebbian: Adjust weights to enforce correlations
  - Parameters: weights w_{ij}
  - Binary output = Θ(weighted sum of inputs)
    - Take w_o to be the threshold with fixed input –1.

$$Output_i = \Theta\left(\sum_j w_{ij}\xi_j\right)$$

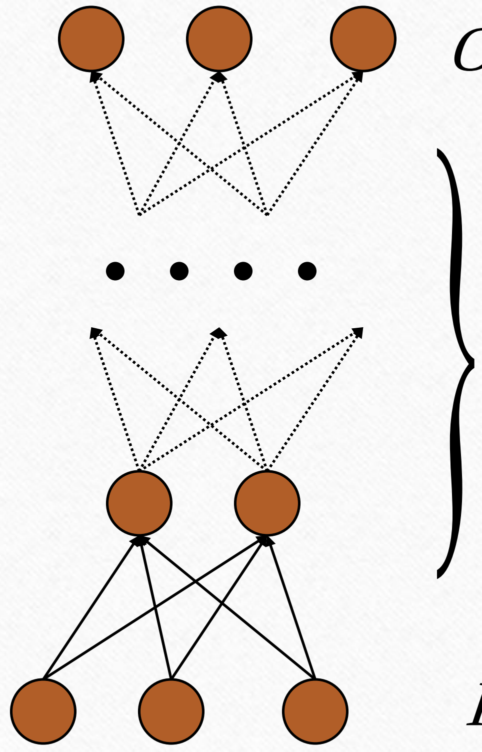


Single-layer



Multilayer

Multilayer Perceptron



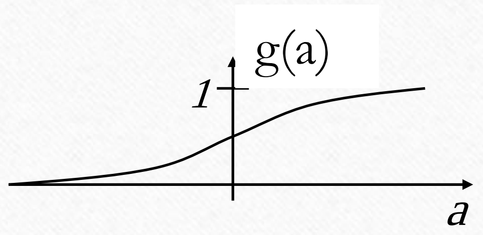
Output neurons

One or more layers of hidden units (hidden layers)

Input nodes

The most common output function (Sigmoid):

$$g(a) = \frac{1}{1 + e^{-\beta a}}$$



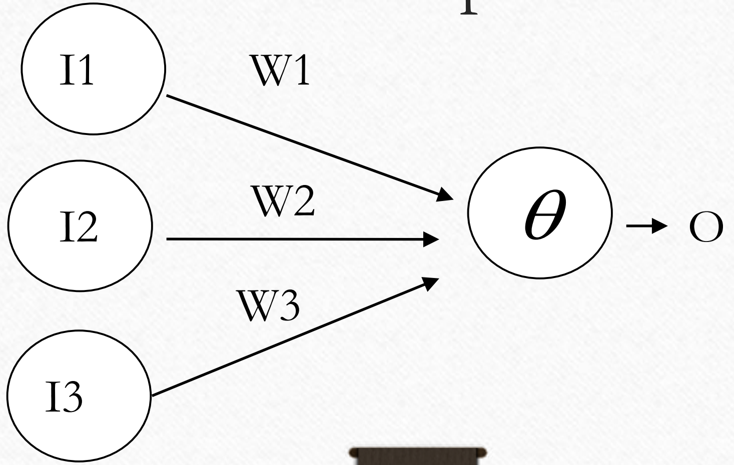
(non-linear squashing function)

Perceptrons

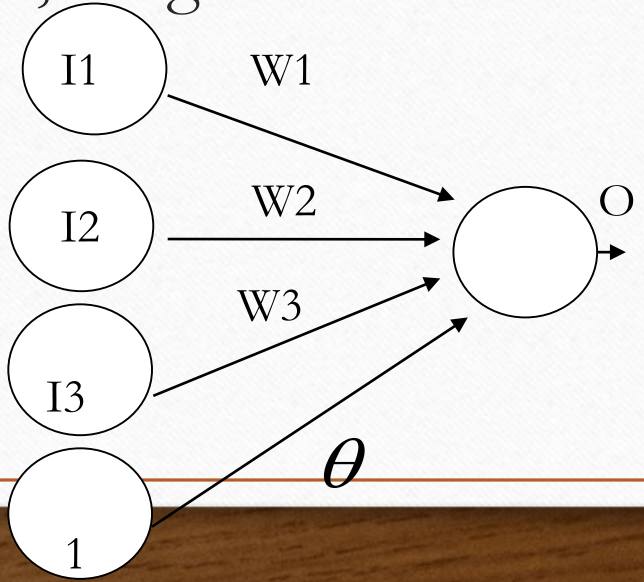
- Initial proposal of connectionist networks
- Rosenblatt, 50's and 60's
- Essentially a linear discriminant composed of nodes, weights

Activation Function

$$O = \begin{cases} 1 : \left(\sum_i w_i I_i \right) + \theta > 0 \\ 0 : otherwise \end{cases}$$



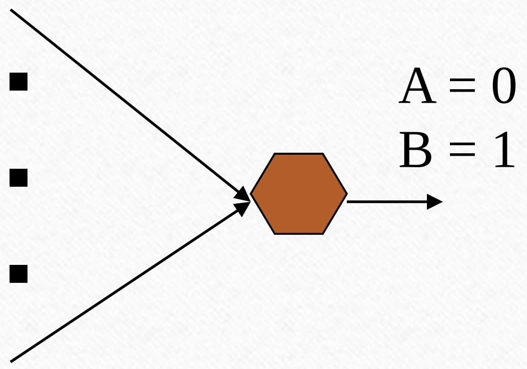
or

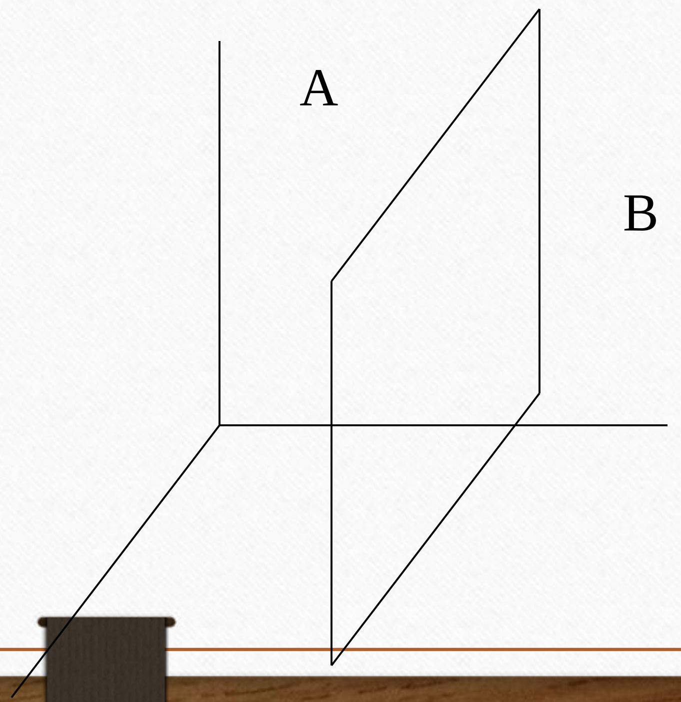


Single layer Perceptron

Single Layer of Neurons

A single neuron with unit step activation function can classify the input into two categories





Single layer Perceptron

Single Layer of Neurons

However, we can also use one neuron to classify only one class. The neuron decides whether the input belongs to its class or not

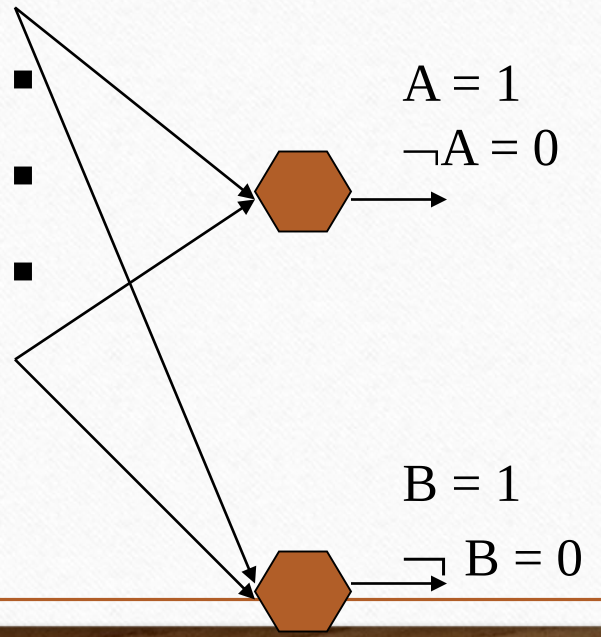
This configuration has the disadvantage that the network size become large

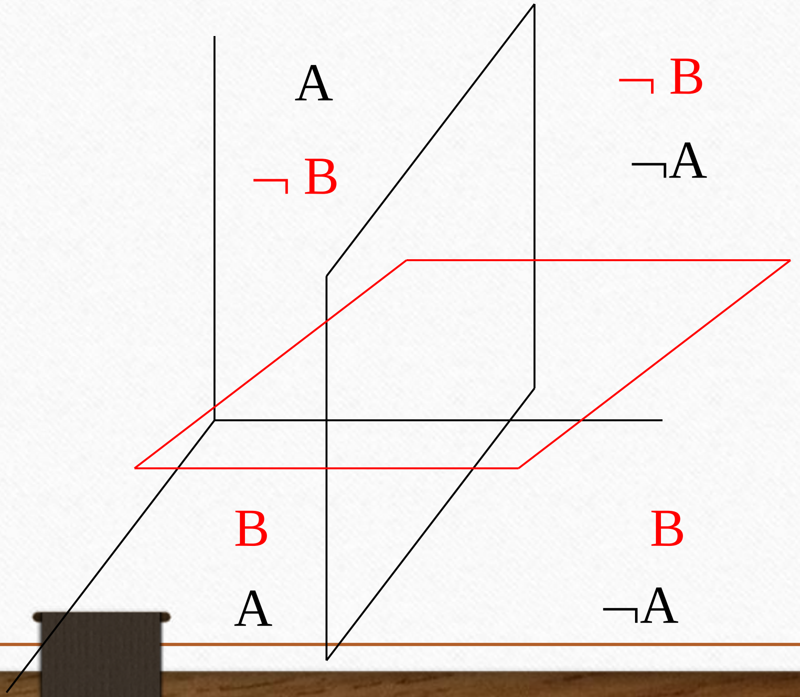
However, it has the advantage that an input may be placed in more than one class, or in none of the classes.

Single layer Perceptron

Single Layer of Neurons

Two neurons for two categories

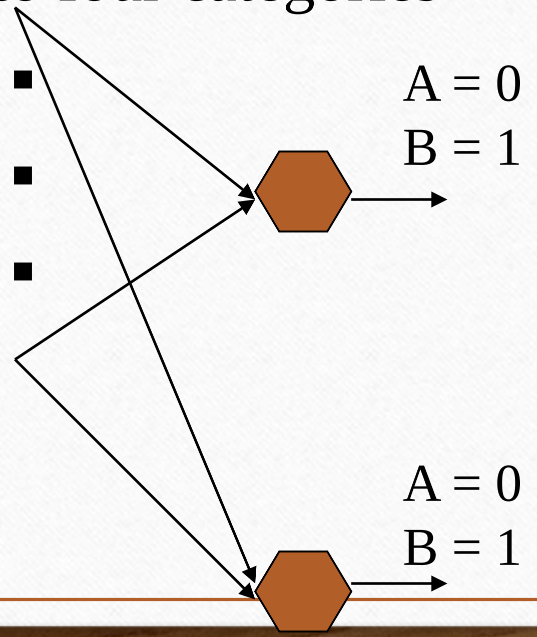


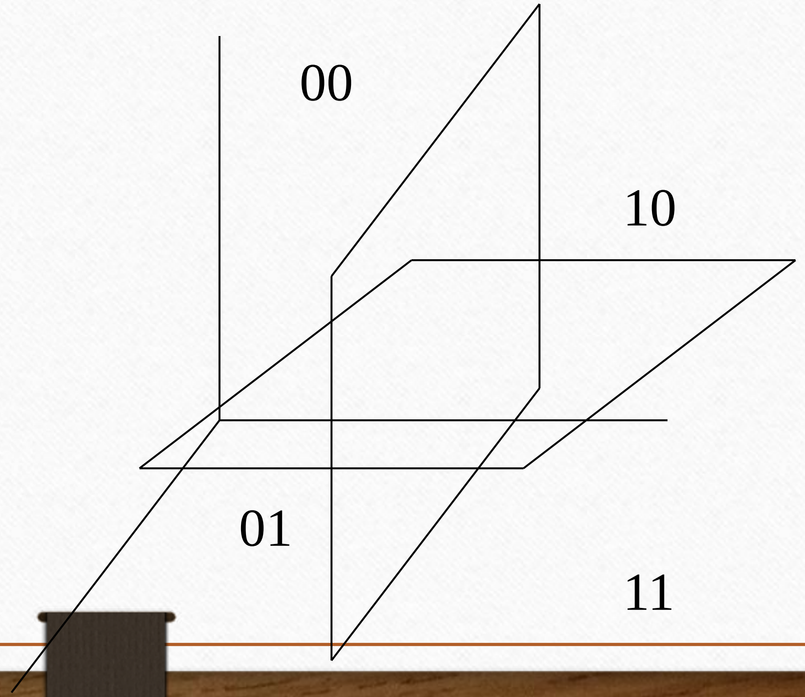


Single layer Perceptron

Single Layer of Neurons

Two neurons with unit step activation function can classify the input into four categories





Perceptron Example

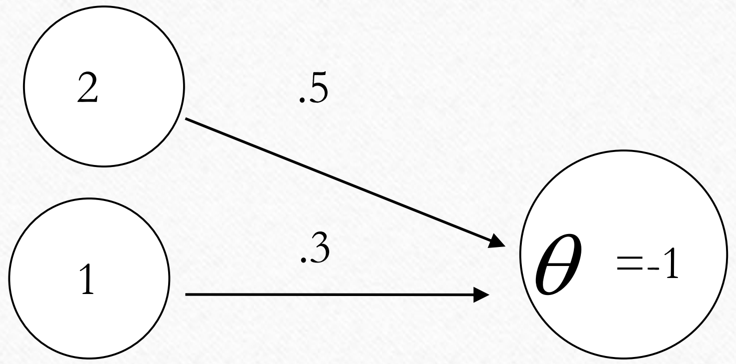
Learning Procedure:

Randomly assign weights (between 0-1)

Present inputs from training data

Get output O, nudge weights to gives results toward our desired output T

Repeat; stop when no errors, or enough epochs completed



2(0.5) + 1(0.3) + -1 = 0.3 , O=1

Perception Training

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

$$\Delta w_i(t) = (T - O)I_i$$

Weights include Threshold. T=Desired, O=Actual output.

Example: T=0, O=1, W1=0.5, W2=0.3, I1=2, I2=1,Theta=-1

$$w_1(t+1) = 0.5 + (0-1)(2) = -1.5$$
$$w_2(t+1) = 0.3 + (0-1)(1) = -0.7$$
$$w_\theta(t+1) = -1 + (0-1)(1) = -2$$

If we present this input again, we'd output 0 instead

How might you use a perceptron network?

- This (and other networks) are generally used to learn how to make classifications
- Say you have collected some data regarding the diagnosis of patients with heart disease
  - Age, Sex, Chest Pain Type, Resting BPS, Cholesterol, …, Diagnosis (<50% diameter narrowing, >50% diameter narrowing)
  - `67,1,4,120,229,…, 1`
  - `37,1,3,130,250,… ,0`
  - `41,0,2,130,204,… ,0`
- Train network to predict heart disease of new patient

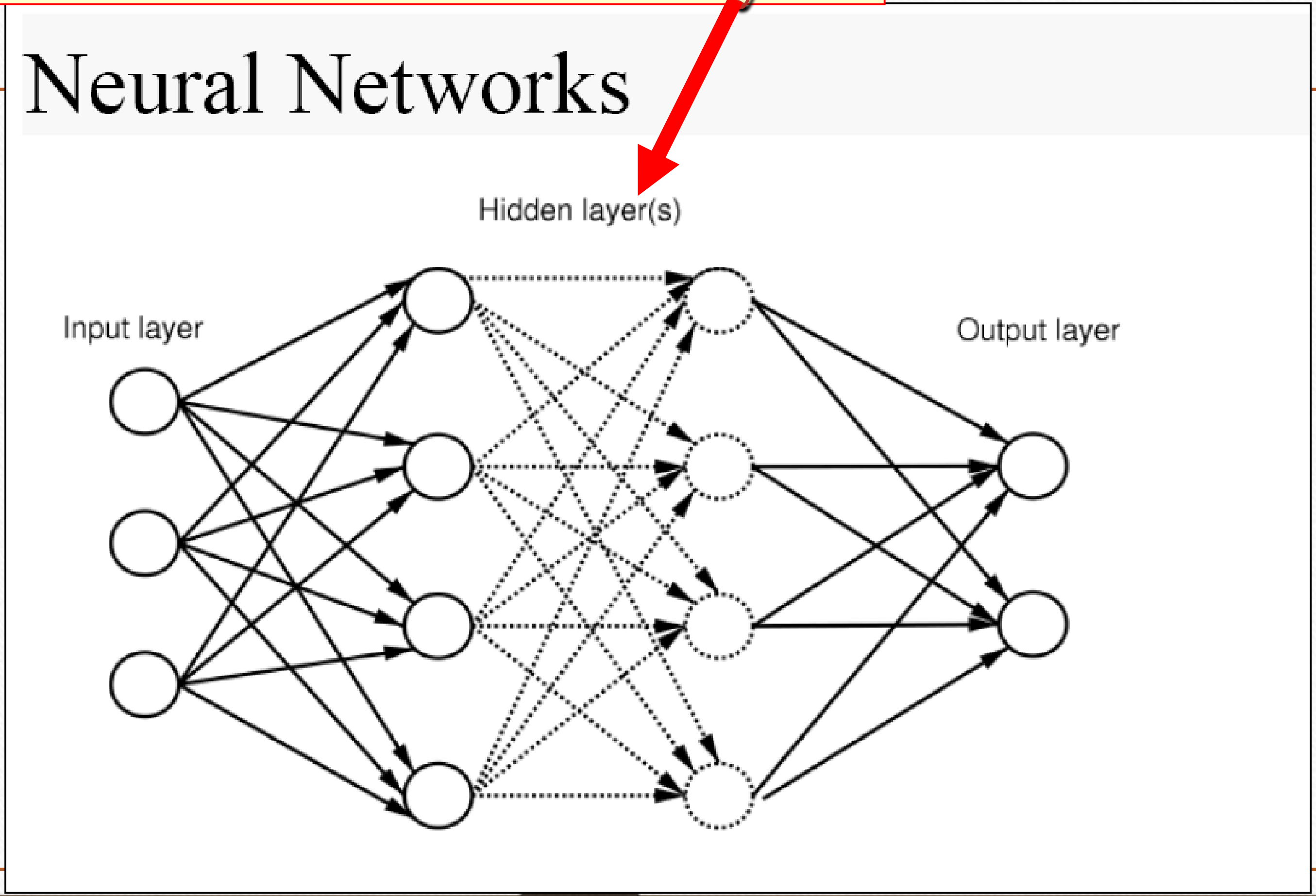
Multilayer Perceptron

N-layer FeedForward Network

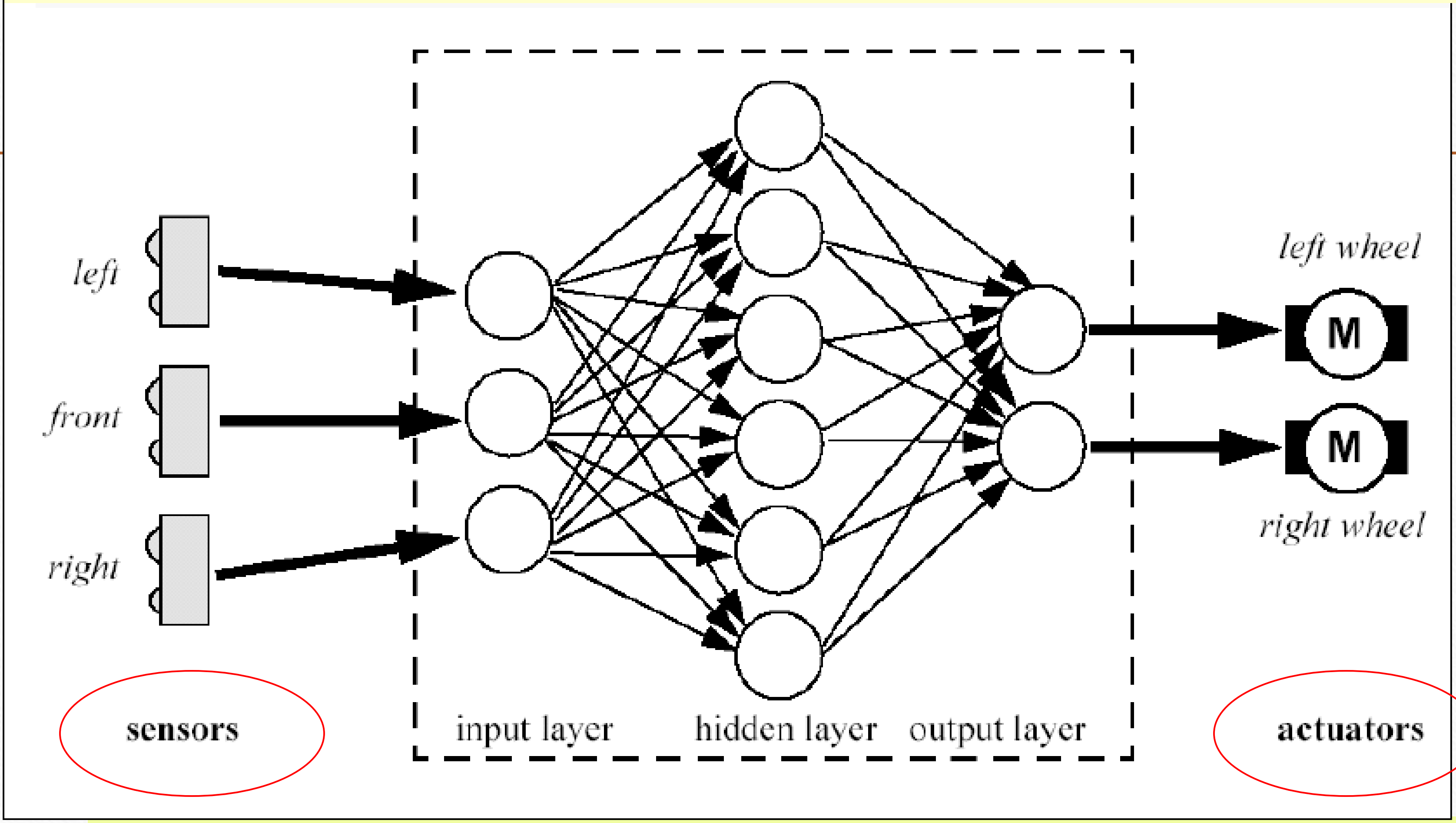
- Layer 0 is input nodes
- Layers 1 to N-1 are hidden nodes
- Layer N is output nodes
- All nodes at any layer k are connected to all nodes at layer k+1
- There are no cycles

Multilayer Perceptron

Feed-forward NN with hidden layer



Reactive architecture based on NN for a simple robot



- Braitenberg Vehicles
- Quantum Neural BV

Multilayer Perceptron

Evaluation of a ***Feedforward NN*** using software is easy

Set bias input neuron

```
void feedforward(float N_in[NIN],float N_hid[NHID],float N_out[NOUT])
{ int i,j;
  N_in[NIN-1] = 1.0; // set bias input neuron
  for (i=0; i<NHID-1; i++) // calculate activation of hidden neurons
  { N_hid[i] = 0.0;
    for (j=0; j<NIN; j++)
      N_hid[i] += N_in[j] * w_in[j][i];
    N_hid[i] = sigmoid(N_hid[i]);
  }
  N_hid[NHID-1] = 1.0; // set bias hidden neuron
  for (i=0; i<NOUT; i++) // calculate output neurons
  { N_out[i] = 0.0;
    for (j=0; j<NHID; j++)
      N_out[i] += N_hid[j] * w_out[j][i];
    N_out[i] = sigmoid(N_out[i]);
  }
}
```

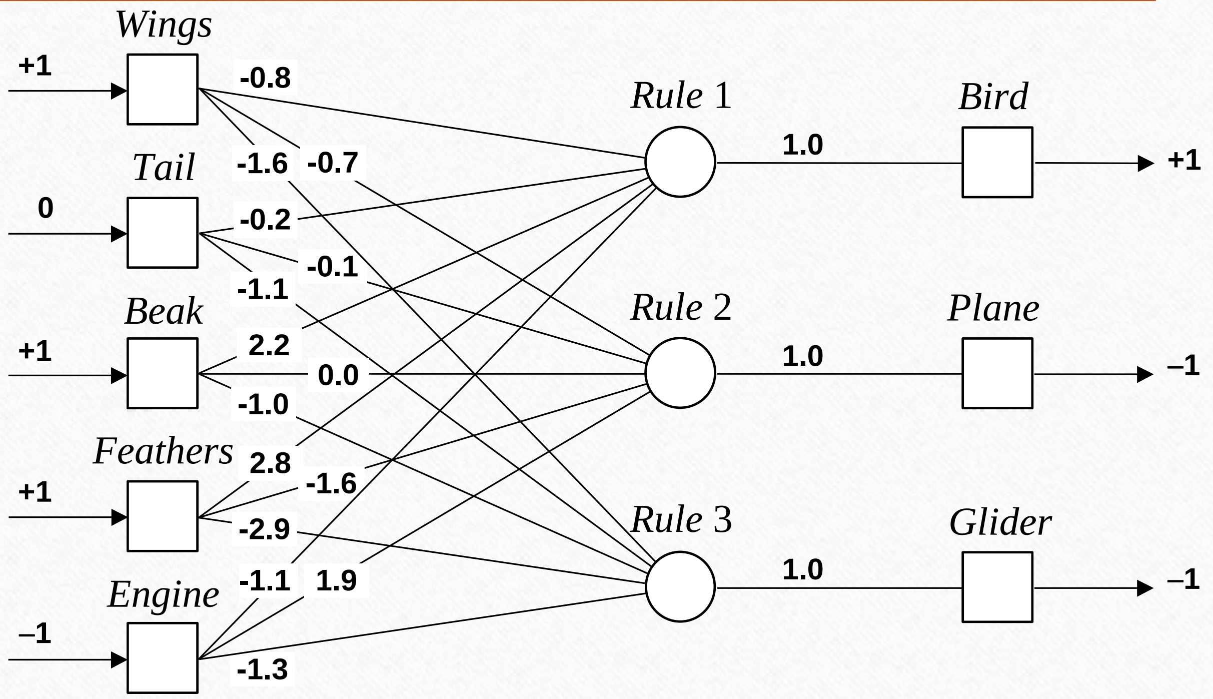
Calculate activation of hidden neurons

Calculate output neurons

Take from hidden neurons and multiply by weights

Rule Extraction

- Neurons in the network are connected by links, each of which has a numerical weight attached to it.
- The weights in a trained neural network determine the strength or importance of the associated neuron inputs.

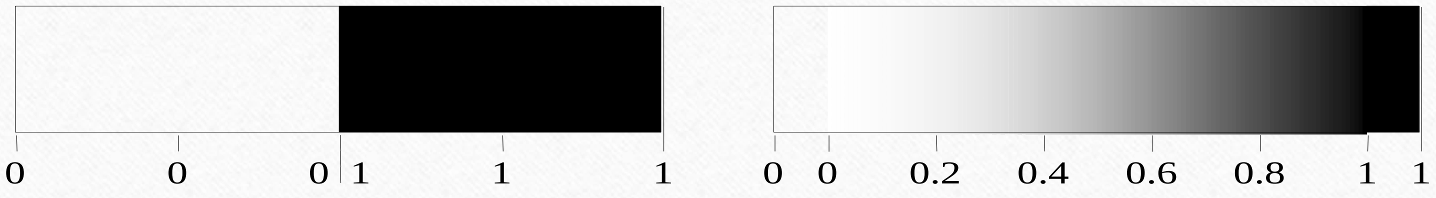


Neuro-Fuzzy Systems

- Integrated neuro-fuzzy systems can combine the parallel computation and learning abilities of neural networks with the human-like knowledge representation and explanation abilities of fuzzy systems.
- As a result, neural networks become more transparent, while fuzzy systems become capable of learning.
- A neuro-fuzzy system is a neural network which is functionally equivalent to a fuzzy inference model. It can be trained to develop IF-THEN fuzzy rules and determine membership functions for input and output variables of the system.
- The connectionist structure avoids fuzzy inference, which entails a substantial computational burden.

More Definitions

- Fuzzy logic is a set of mathematical principles for knowledge representation based on **degrees of membership**.
- Unlike two-valued Boolean logic, fuzzy logic is **multi-valued**. It deals with **degrees of membership** and **degrees of truth**.
- Fuzzy logic uses the continuum of logical values between 0 (completely false) and 1 (completely true). Instead of just black and white, it employs the spectrum of colours, accepting that things can be partly true and partly false at the same time.

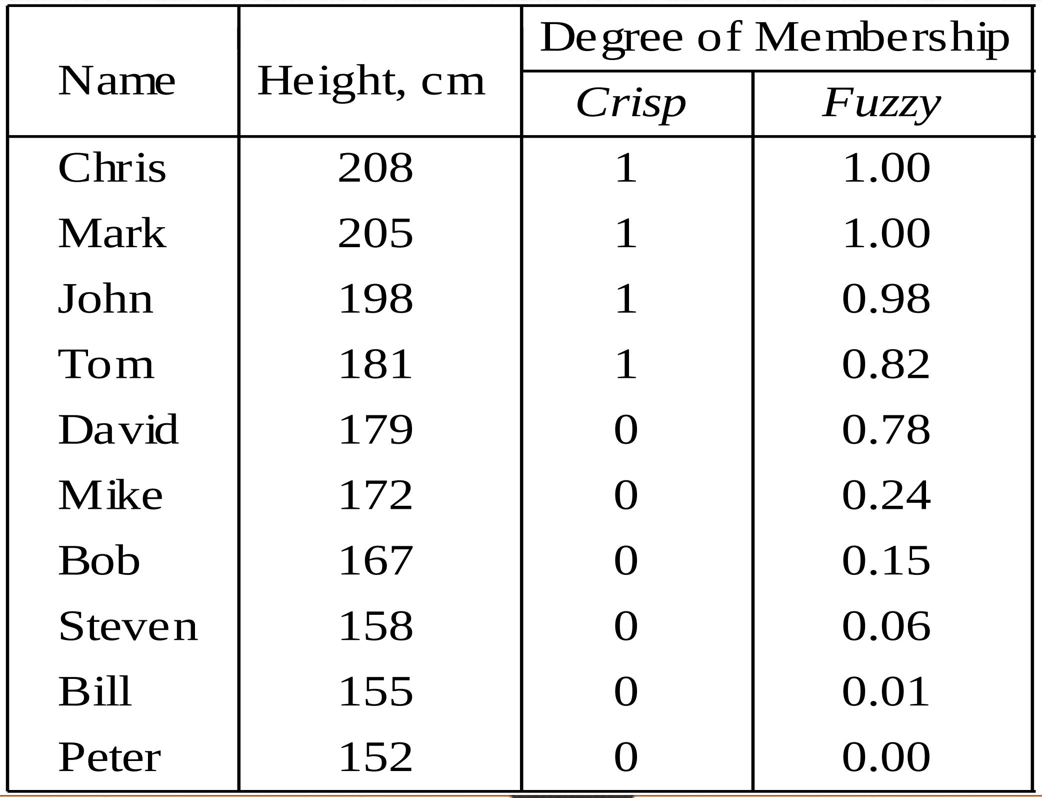


(*a*) Boolean Logic. (*b*) Multi-valued Logic.

Fuzzy Sets

- The concept of a **set** is fundamental to mathematics.
- However, our own language is also the supreme expression of sets. For example, *car* indicates the *set of cars*. When we say a car, we mean one out of the set of cars.
- The classical example in fuzzy sets is tall men. The elements of the fuzzy set “tall men” are all men, but their degrees of membership depend on their height. (see table on next page)

Fuzzy Sets

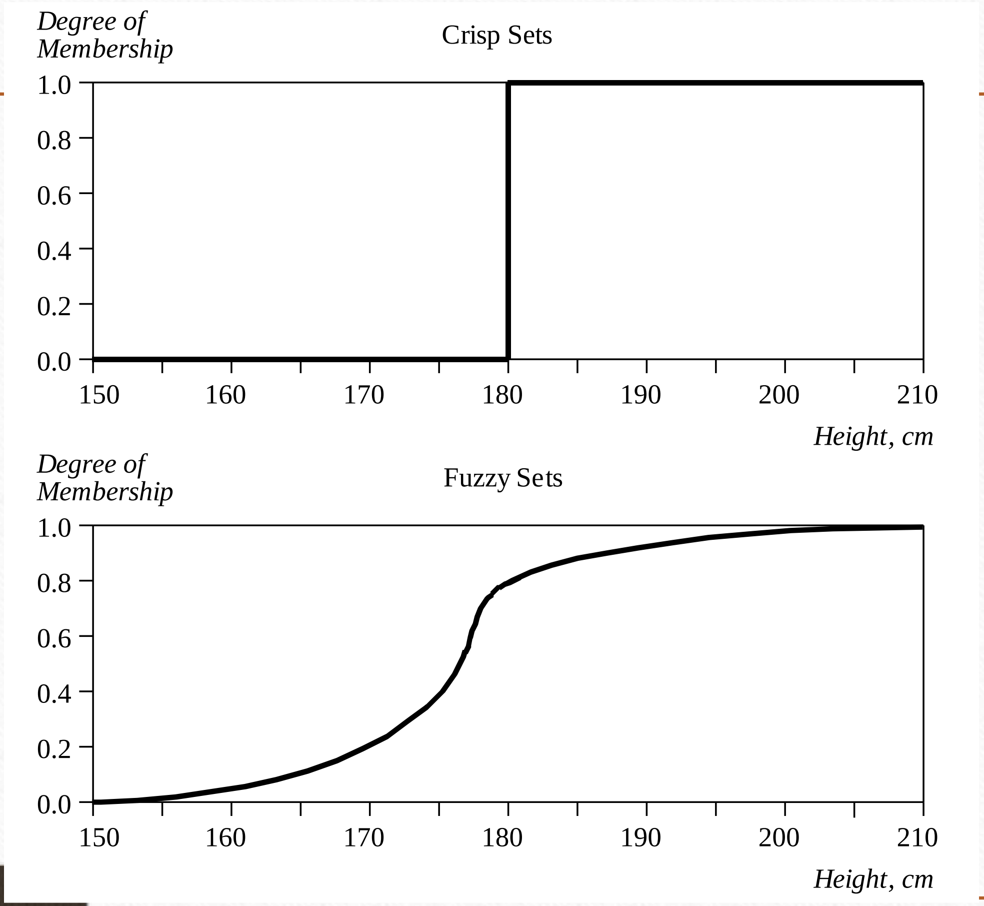


Name	Height, cm	Degree of Membership	
		Crisp	*Fuzzy*
Chris	208	1	1.00
Mark	205	1	1.00
John	198	1	0.98
Tom	181	1	0.82
David	179	0	0.78
Mike	172	0	0.24
Bob	167	0	0.15
Steven	158	0	0.06
Bill	155	0	0.01
Peter	152	0	0.00

Crisp Vs Fuzzy Sets

The x-axis represents the **universe of discourse** – the range of all possible values applicable to a chosen variable. In our case, the variable is the man height. According to this representation, the universe of men's heights consists of all tall men.

The y-axis represents the **membership value of the fuzzy set**. In our case, the fuzzy set of "tall men" maps height values into corresponding membership values.

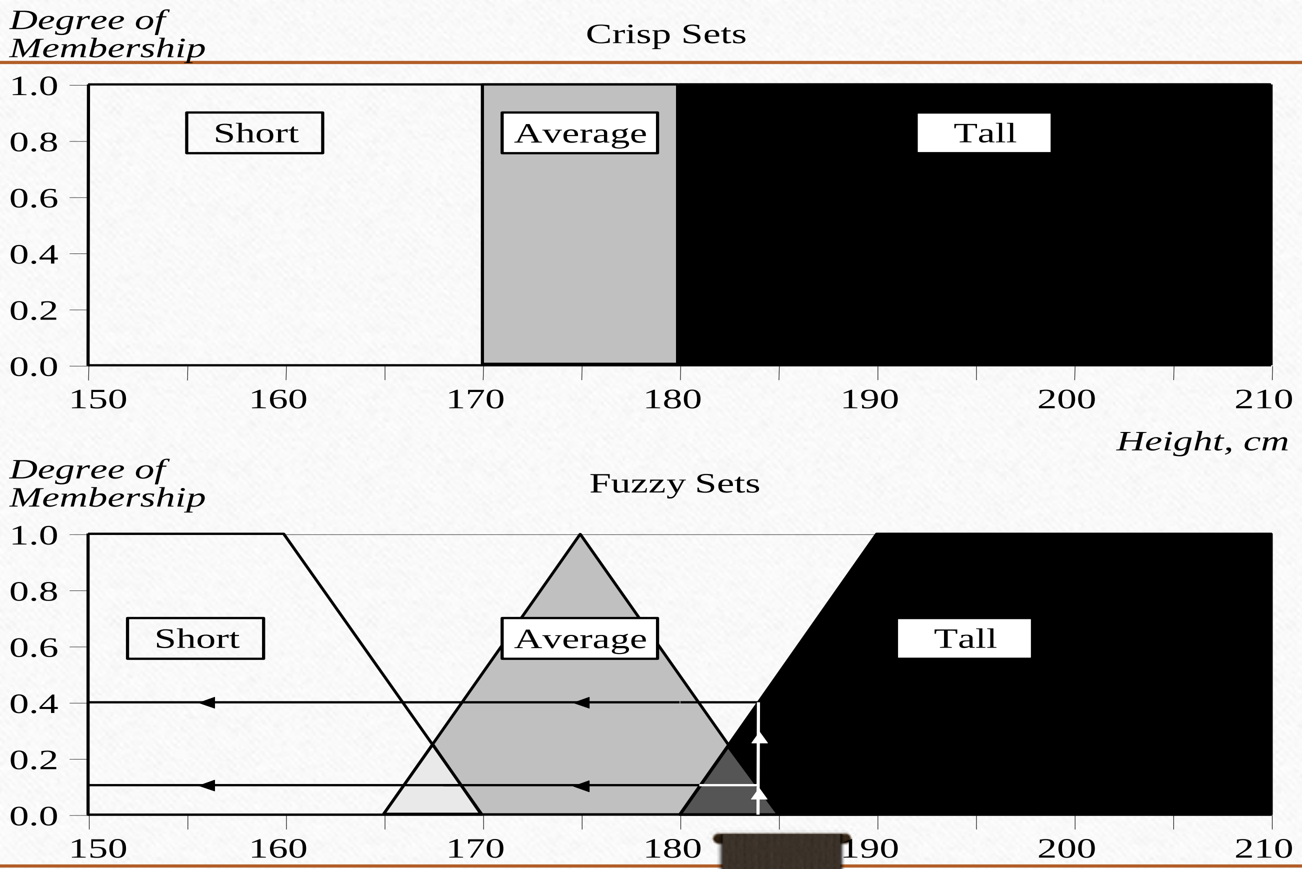


Fuzzy Set Representation

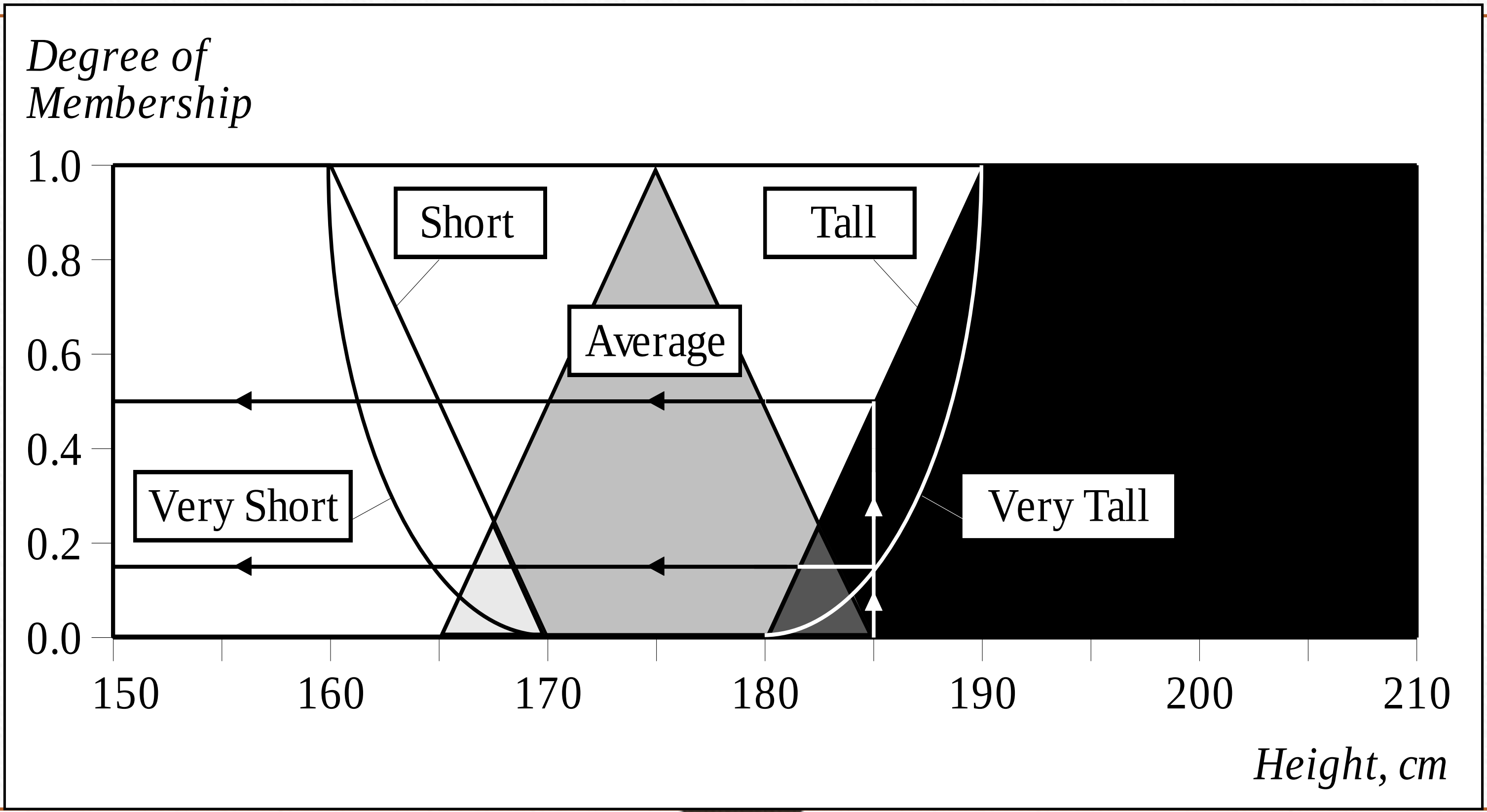
- First, we determine the membership functions. In our “tall men” example, we can obtain fuzzy sets of *tall*, *short* and *average* men.

- The universe of discourse – the men’s heights – consists of three sets: *short*, *average* and *tall* men. As you will see, a man who is 184 cm tall is a member of the *average* men set with a degree of membership of 0.1, and at the same time, he is also a member of the *tall* men set with a degree of 0.4. (see graph on next page)

Fuzzy Set Representation



Linguistic Variables and Hedges



Mamdani Fuzzy Inference

- The Mamdani-style fuzzy inference process is performed in four steps:

1. Fuzzification of the input variables
2. Rule evaluation (inference)
3. Aggregation of the rule outputs (composition)
4. Defuzzification.

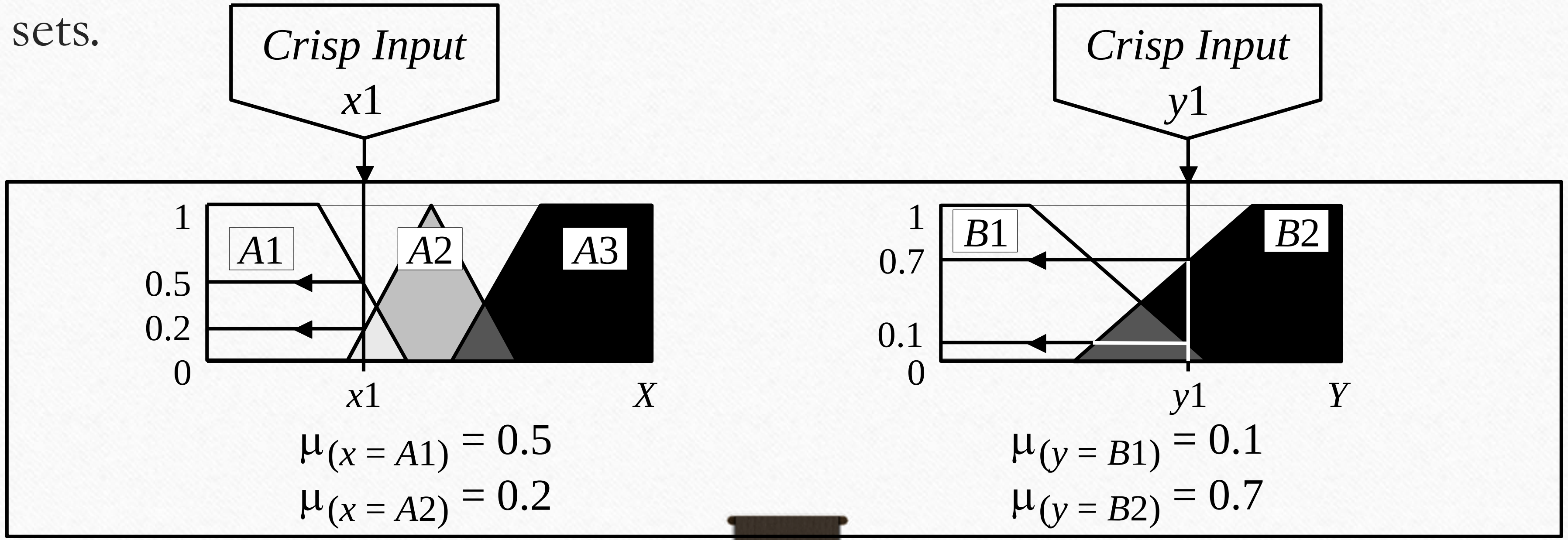
Mamdani Fuzzy Inference

We examine a simple two-input one-output problem that includes three rules:

Rule: 1		Rule: 1		
IF	x is A3	IF	project_funding	is adequate
OR	y is B1	OR	project_staffing	is small
THEN	z is C1	THEN	risk	is low
Rule: 2		Rule: 2		
IF	x is A2	IF	project_funding	is marginal
AND	y is B2	AND	project_staffing	is large
THEN	z is C2	THEN	risk	is normal
Rule: 3		Rule: 3		
IF	x is A1	IF	project_funding	is inadequate
THEN	z is C3	THEN	risk	is high

Step 1: Fuzzification

- The first step is to take the crisp inputs, x1 and y1 (*project funding* and *project staffing*), and determine the degree to which these inputs belong to each of the appropriate fuzzy sets.



$\mu_{(x = A1)} = 0.5$
$\mu_{(x = A2)} = 0.2$
$\mu_{(y = B1)} = 0.1$
$\mu_{(y = B2)} = 0.7$

Step 2: Rule Evaluation

- The second step is to take the fuzzified inputs, $\mu_{(x=A1)} = 0.5$, $\mu_{(x=A2)} = 0.2$, $\mu_{(y=B1)} = 0.1$ and $\mu_{(y=B2)} = 0.7$, and apply them to the antecedents of the fuzzy rules.
- If a given fuzzy rule has multiple antecedents, the fuzzy operator (AND or OR) is used to obtain a single number that represents the result of the antecedent evaluation.
- This number (the truth value) is then applied to the consequent membership function.

Fuzzy Sets Example

- Air-conditioning involves the delivery of air which can be warmed or cooled and have its humidity raised or lowered.
- An air-conditioner is an apparatus for controlling, especially lowering, the temperature and humidity of an enclosed space. An air-conditioner typically has a fan which blows/cools/circulates fresh air and has cooler and the cooler is under thermostatic control. Generally, the amount of air being compressed is proportional to the ambient temperature.
- Consider Johnny’s air-conditioner which has five control switches: COLD, COOL, PLEASANT, WARM and HOT. The corresponding speeds of the motor controlling the fan on the air-conditioner has the graduations: MINIMAL, SLOW, MEDIUM, FAST and BLAST.

Fuzzy Sets Example

- The rules governing the air-conditioner are as follows:

RULE 1:

IF TEMP is COLD THEN SPEED is MINIMAL

RULE 2:

IF TEMP is COOL THEN SPEED is SLOW

RULE 3:

IF TEMP is PLEASANT THEN SPEED is MEDIUM

RULE 4:

IF TEMP is WARM THEN SPEED is FAST

RULE 5:

IF TEMP is HOT THEN SPEED is BLAST

Fuzzy Sets Example

The **temperature** graduations are related to Johnny's perception of ambient temperatures.

where:

Y : *temp* value belongs to the set ($0<\mu_A(x)<1$)

Y* : *temp* value is the ideal member to the set ($\mu_A(x)=1$)

N : temp value is not a member of the set ($\mu_A(x)=0$)

Temp (^{0}C).	COLD	COOL	PLEASANT	WARM	HOT
0	Y*	N	N	N	N
5	Y	Y	N	N	N
10	N	Y	N	N	N
12.5	N	Y*	N	N	N
15	N	Y	N	N	N
17.5	N	N	Y*	N	N
20	N	N	N	Y	N
22.5	N	N	N	Y*	N
25	N	N	N	Y	N
27.5	N	N	N	N	Y
30	N	N	N	N	Y*

Fuzzy Sets Example

Johnny's perception of the **speed** of the motor is as follows:

where:

Y : *temp* value belongs to the set ($0<\mu_A(x)<1$)

Y* : *temp* value is the ideal member to the set ($\mu_A(x)=1$)

N : temp value is not a member of the set ($\mu_A(x)=0$)

Rev/sec (RPM)	MINIMAL	SLOW	MEDIUM	FAST	BLAST
0	Y*	N	N	N	N
10	Y	N	N	N	N
20	Y	Y	N	N	N
30	N	Y*	N	N	N
40	N	Y	N	N	N
50	N	N	Y*	N	N
60	N	N	N	Y	N
70	N	N	N	Y*	N
80	N	N	N	Y	Y
90	N	N	N	N	Y
100	N	N	N	N	Y*

Fuzzy Sets Example

- The analytically expressed membership for the reference fuzzy subsets for the **temperature** are:

- COLD:

  for $0 \leq t \leq 10$ $\mu_{COLD}(t) = -t / 10 + 1$

- SLOW:

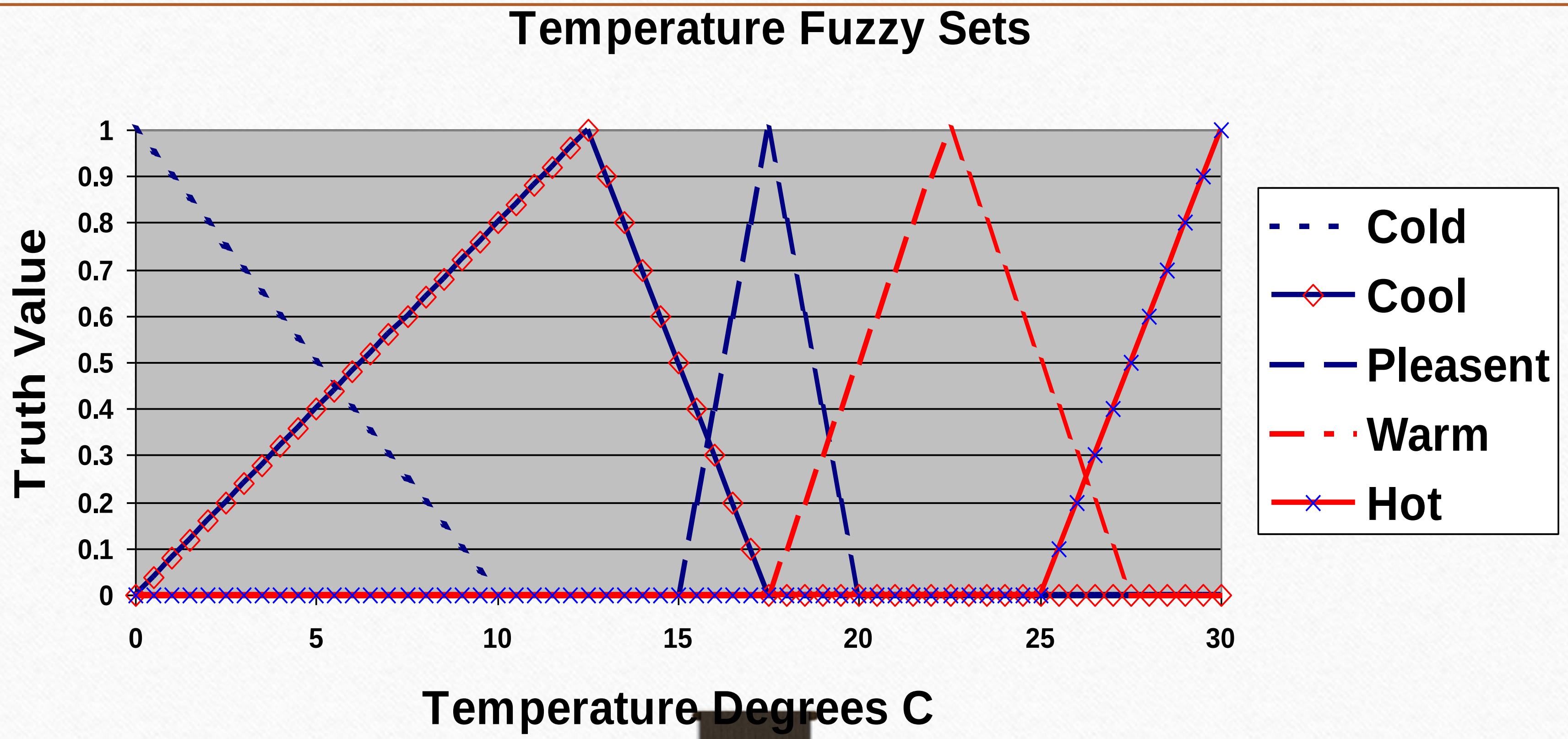
  for $0 \leq t \leq 12.5$ $\mu_{SLOW}(t) = t / 12.5$

  for $12.5 \leq t \leq 17.5$ $\mu_{SLOW}(t) = -t / 5 + 3.5$

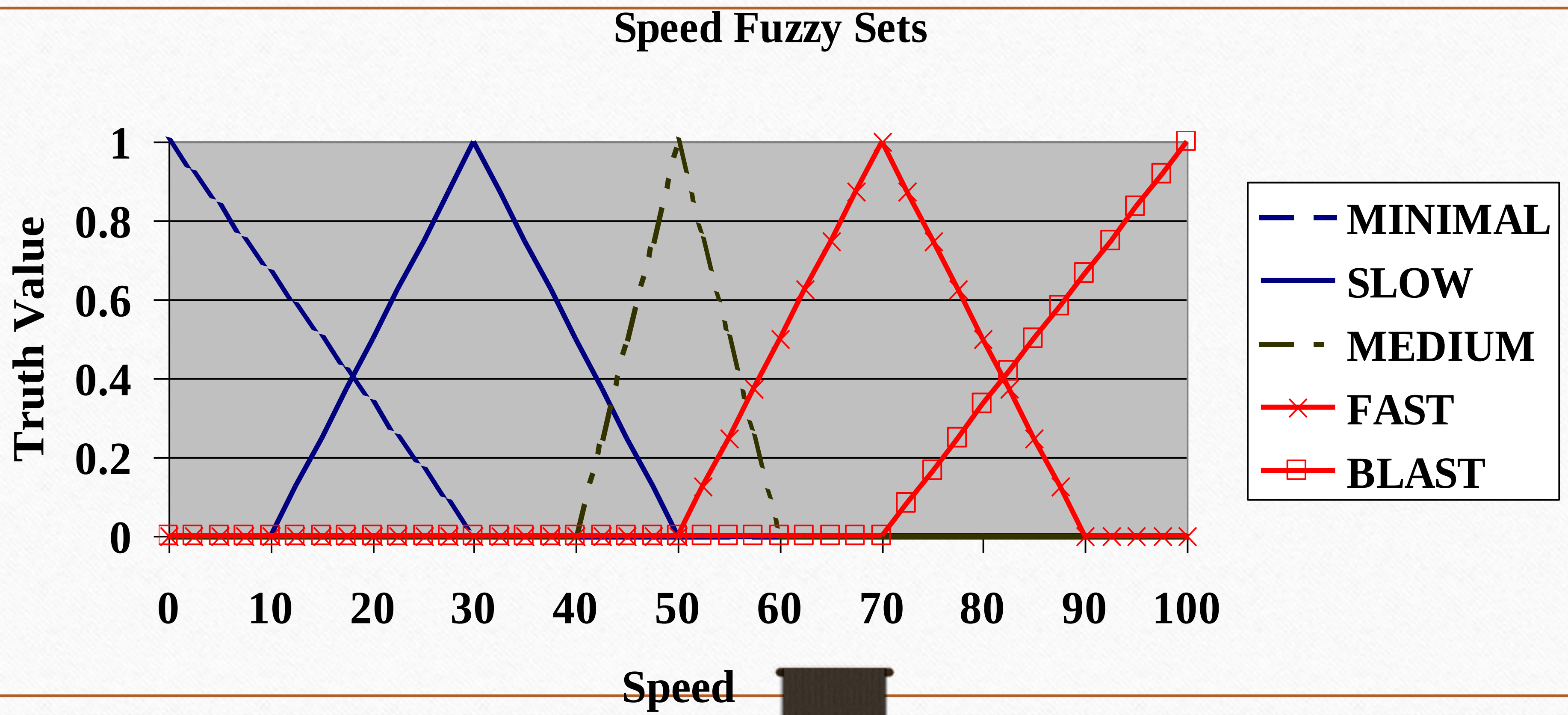
- etc… all based on the linear equation:

  $y = ax + b$

Fuzzy Sets Example



Fuzzy Sets Example



Exercises

For

$A = \{0.2/a, 0.4/b, 1/c, 0.8/d, 0/e\}$

$B = \{0/a, 0.9/b, 0.3/c, 0.2/d, 0.1/e\}$

Draw the Fuzzy Graph of A and B

Then, calculate the following:

- Support, Core, Cardinality, and Complement for A and B independently
- Union and Intersection of A and B
- the new set C, if $C = A^2$
- the new set D, if $D = 0.5 \times B$
- the new set E, for an alpha cut at $A_{0.5}$

Solutions

A = {0.2/a, 0.4/b, 1/c, 0.8/d, 0/e}

B = {0/a, 0.9/b, 0.3/c, 0.2/d, 0.1/e}

<u>Support</u>

Supp(A) = {a, b, c, d}

Supp(B) = {b, c, d, e}

<u>Core</u>

Core(A) = {c}

Core(B) = {}

<u>Cardinality</u>

Card(A) = 0.2 + 0.4 + 1 + 0.8 + 0 = 2.4

Card(B) = 0 + 0.9 + 0.3 + 0.2 + 0.1 = 1.5

<u>Complement</u>

Comp(A) = {0.8/a, 0.6/b, 0/c, 0.2/d, 1/e}

Comp(B) = {1/a, 0.1/b, 0.7/c, 0.8/d, 0.9/e}

Solutions

A = {*0.2/a, 0.4/b, 1/c, 0.8/d, 0/e*}
B = {*0/a, 0.9/b, 0.3/c, 0.2/d, 0.1/e*}

Union

$A \cup B$ = {0.2/a, 0.9/b, 1/c, 0.8/d, 0.1/e}

Intersection

$A \cap B$ = {0/a, 0.4/b, 0.3/c, 0.2/d, 0/e}

$C=A^2$

C = {0.04/a, 0.16/b, 1/c, 0.64/d, 0/e}

$D = 0.5 \times B$

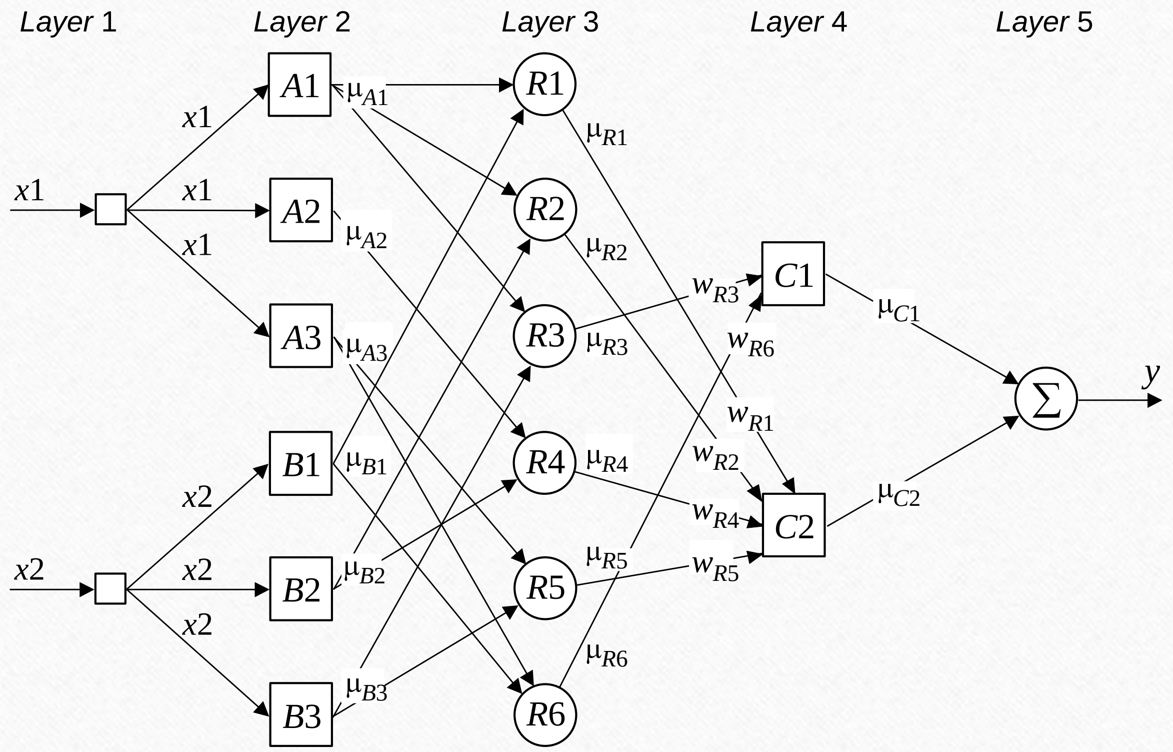
D = {0/a, 0.45/b, 0.15/c, 0.1/d, 0.05/e}

$E = A_{0.5}$

E = {c, d}

Neuro-Fuzzy Systems

- The structure of a neuro-fuzzy system is similar to a multi-layer neural network. In general, a neuro-fuzzy system has input and output layers, and three hidden layers that represent membership functions and fuzzy rules.



Neuro-Fuzzy Systems

- The combination of fuzzy logic and neural networks constitutes a powerful means for designing intelligent systems.

- Domain knowledge can be put into a neuro-fuzzy system by human experts in the form of linguistic variables and fuzzy rules.

- When a representative set of examples is available, a neuro-fuzzy system can automatically transform it into a set of fuzzy IF-THEN rules, and reduce our dependence on expert knowledge when building intelligent systems.

Projects

- abstract: يحدد الفكرة والهدف من المشروع وطريقة تحكم والنتيجة
- introduction and literature survey: يحدد التطبيقات والانواع بالاضافة الى الدراسات التى تناولت نفس الفكرة او افكار مشابهه
- design and control system: يعرض التصميم والحسابات التى اعتمد عليها بالاضافة الى نوع الكنترول المستخدم والخوارزم المعتمد عليه
- experimental test: يعرض تجارب التشغيل سواء للمحاكاه او للنموذج المعملى
- results: يعرض النتائج الخاصة بالمشروع وتحليلها
- conclusion: يتناول ملخص للمشروع من حيث الفكرة والنتائج التى تم التوصل لها
- references: يستعرض قائمة المراجع التى اعتمد عليها المشروع فى جميع البيانات المستخدمه
- *Abbreviations
- بالاضافة الى قائمة الاختصارات التى تم استخدامها فى المشروع

**مطلوب الاتى للمشروع

-technical report
- presentation
- prototype (works)