Computational Intelligence

Dr.N.Santhiyakumari

Professor & Head

Department of ECE

Knowledge Institute of Technology

hod.ece@kiot.ac.in

Computational Intelligence

- **Computational Intelligence (CI)** usually refers to the ability of a computer to learn a specific task from data or experimental observation. Even though it is commonly considered a synonym of soft computing.

Reasons

- **CI** is a set of nature-inspired computational methodologies and approaches to address complex real-world problems to which mathematical or traditional modelling can be useless for a few reasons.

Too complex for mathematical reasoning

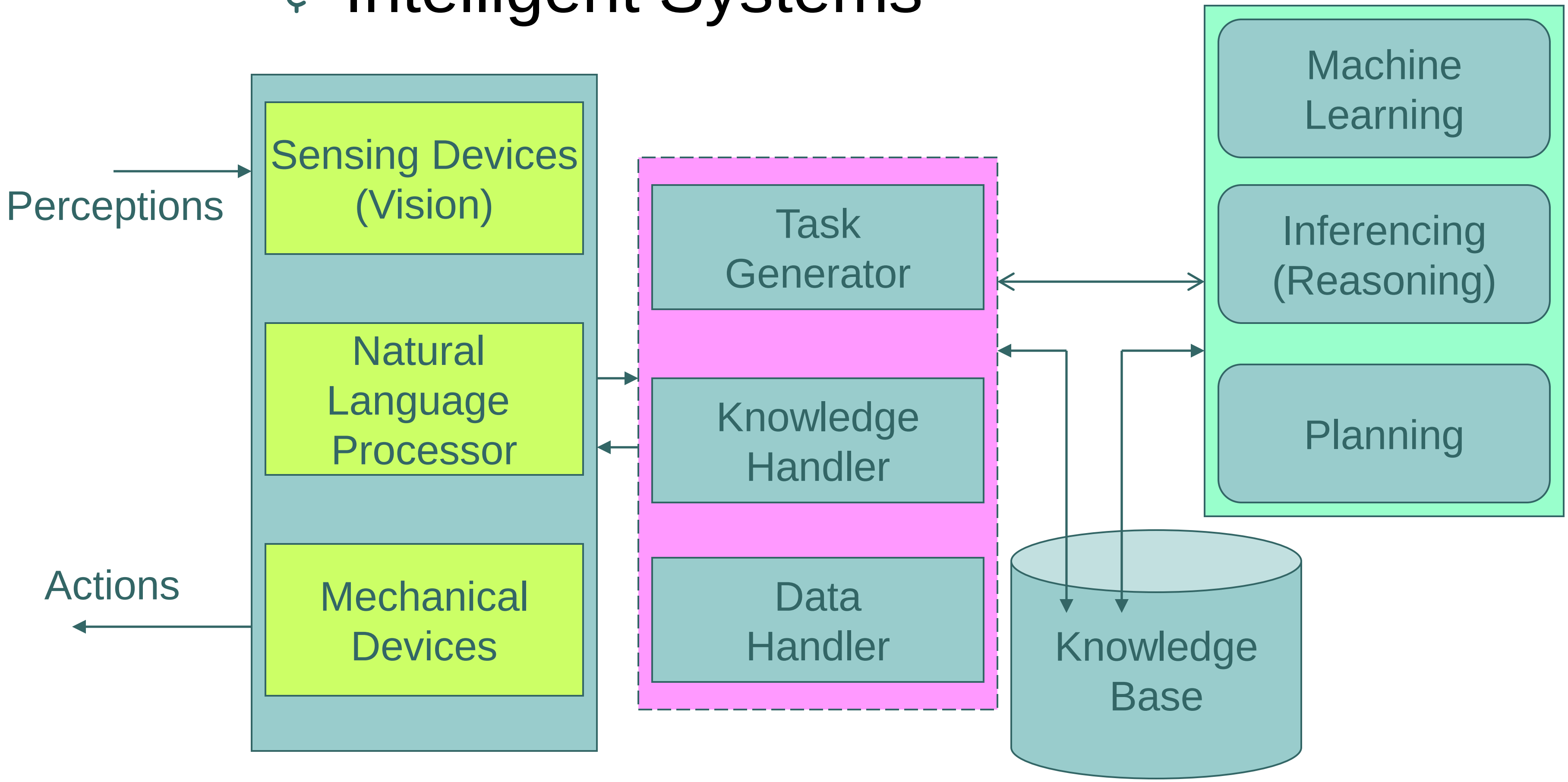
i) **Contain Uncertainties**.

ii) Stochastic in Nature.

iii) **Can't translate into 0 (or) 1.**

From Conventional AI to Computational Intelligence

- Intelligent Systems



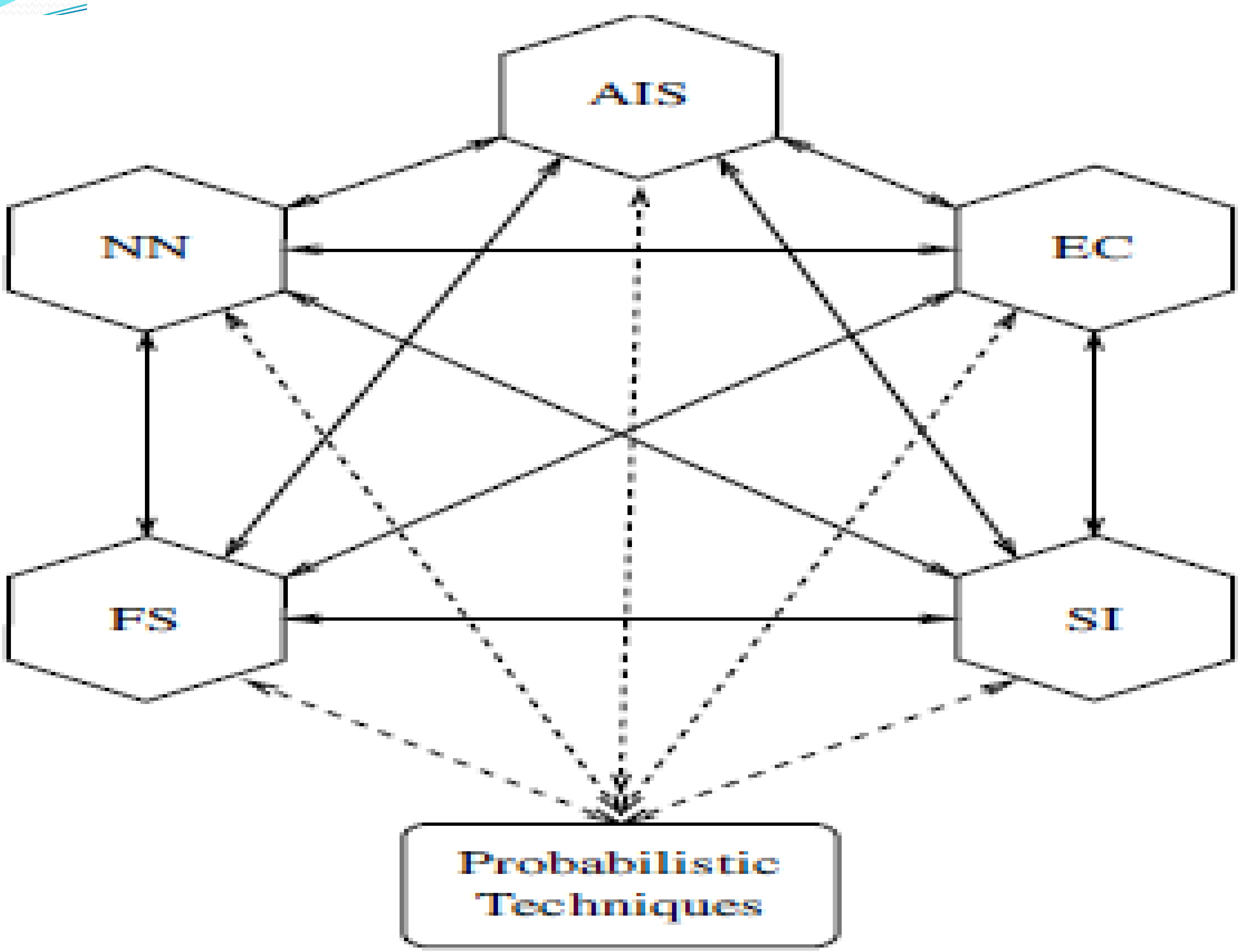
AI – Hard Computing- Binary

CI - Soft Computing - Crisp

CI Techniques

- **1) Artificial neural Networks**
- 2) **Fuzzy Logic**
- **3) Evolutionary Computing**
- 4) **Swarm Intelligence**
- **5) Artificial immune systems**

Computational Intelligence Paradigms



- **WHAT ARE NEURAL NETWORKS?**

1) **Models of the brain and nervous system.**

Information processing paradigm inspired by biological nervous systems, such as our brain.

2) **Highly parallel**

Process information much more like the brain than a serial computer.

3) **Structure:**

Large number of highly interconnected processing elements (*neurons*) working together.

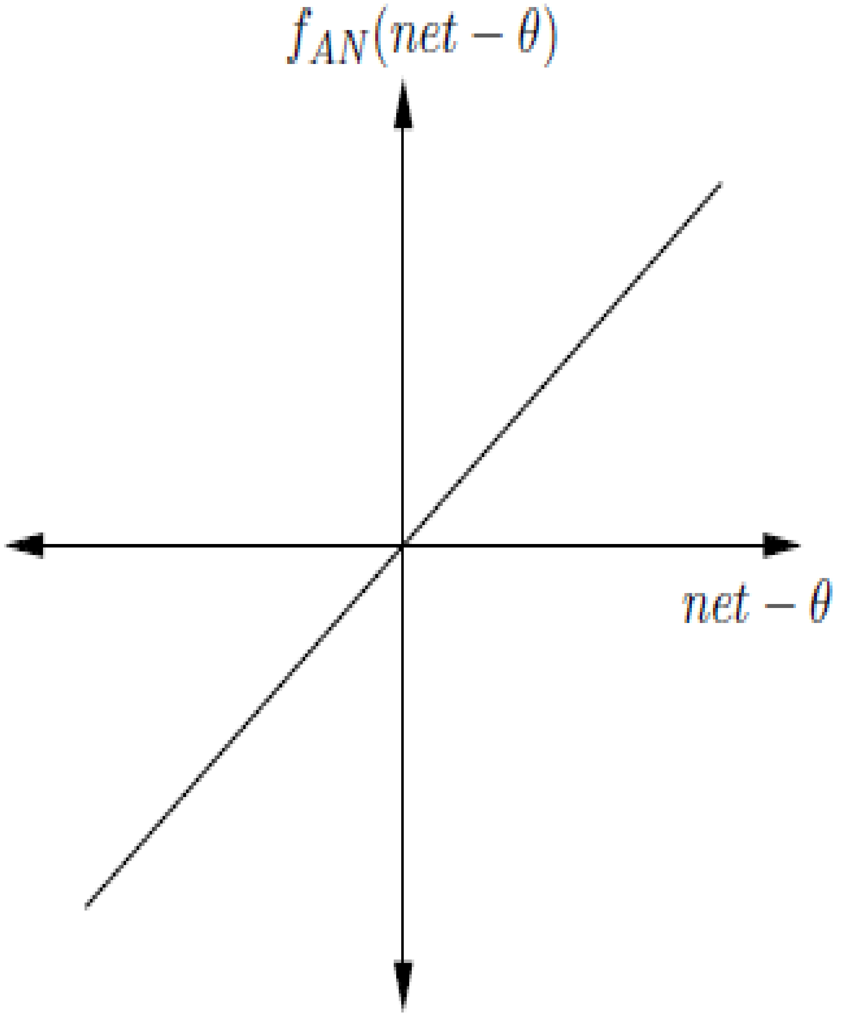
Like people, they learn *from experience* (by example)

Adopts Learning principles

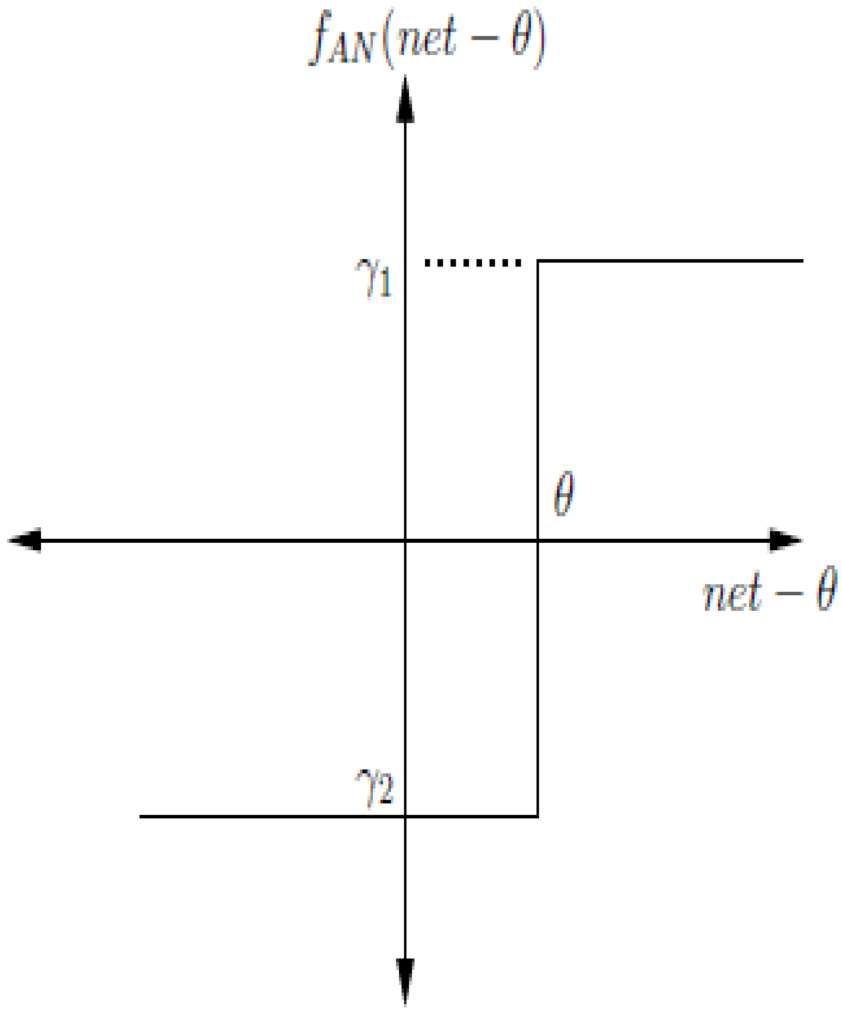
The Brain as an Information Processing System

The human brain contains about 10 billion nerve cells, or **neurons**. On average, each neuron is connected to other neurons through about 10,000 **synapses**. (The actual figures vary greatly, depending on the local neuroanatomy.)

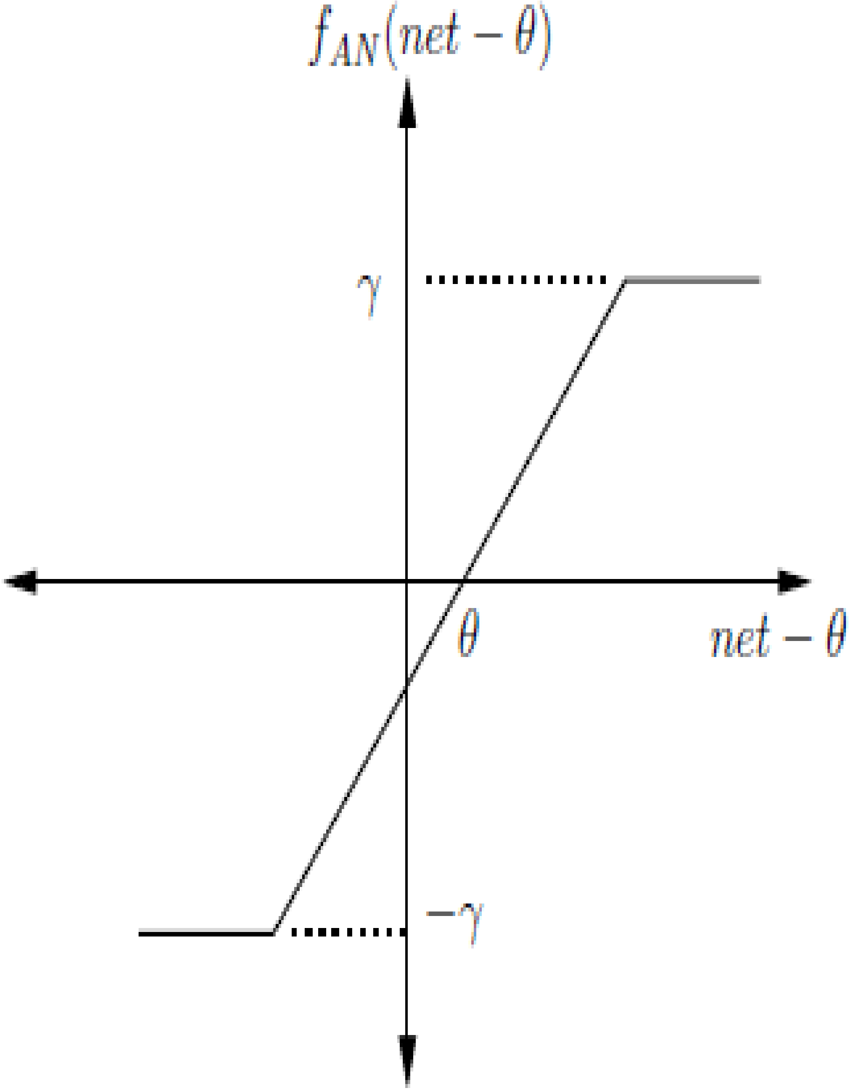
Activation Functions



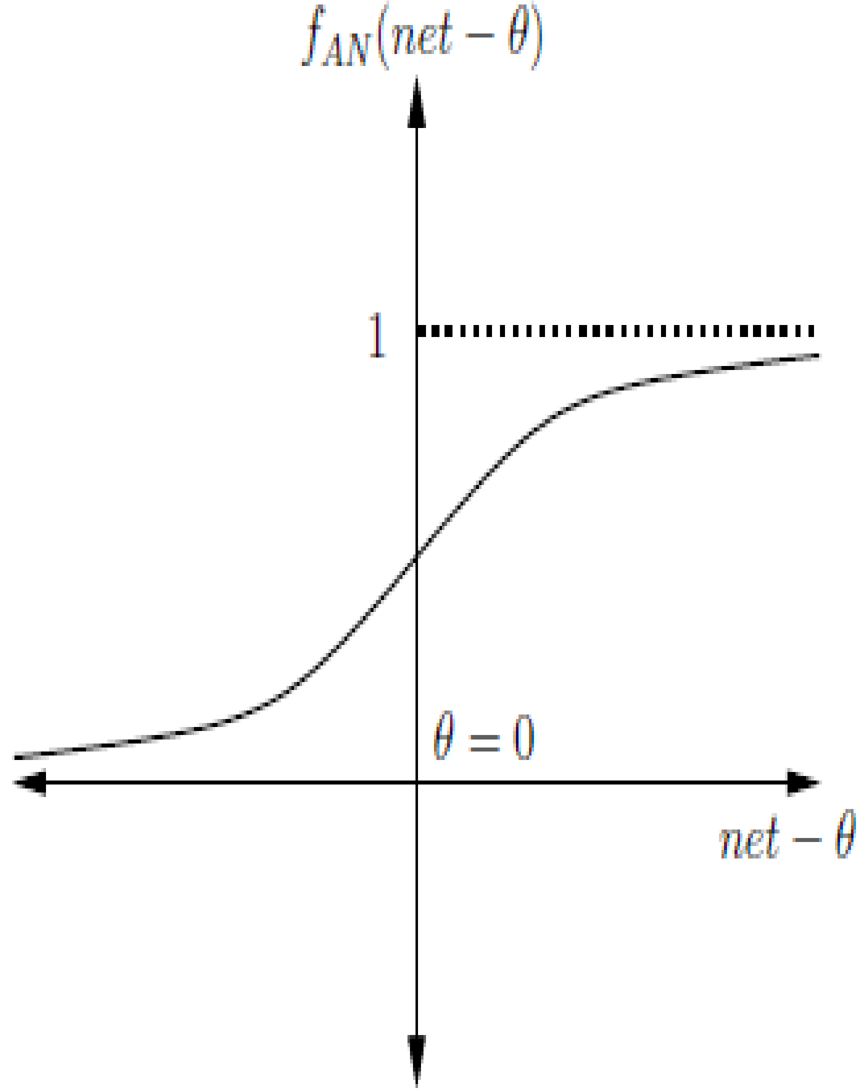
(a) Linear function



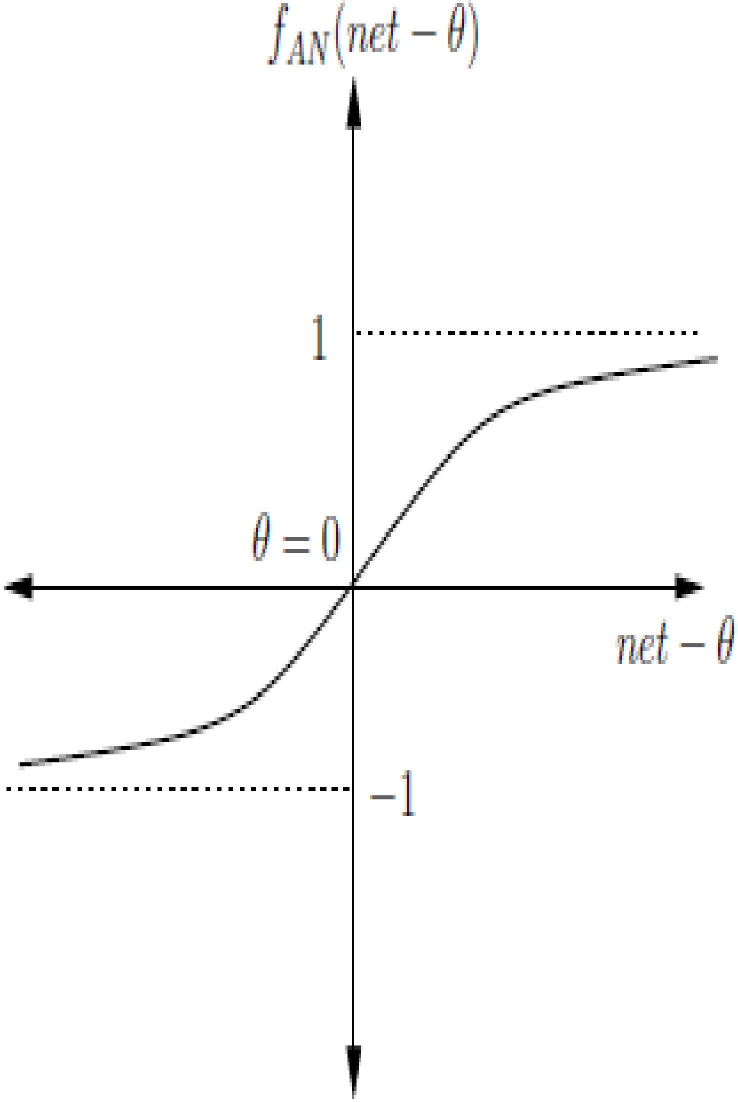
(b) Step function



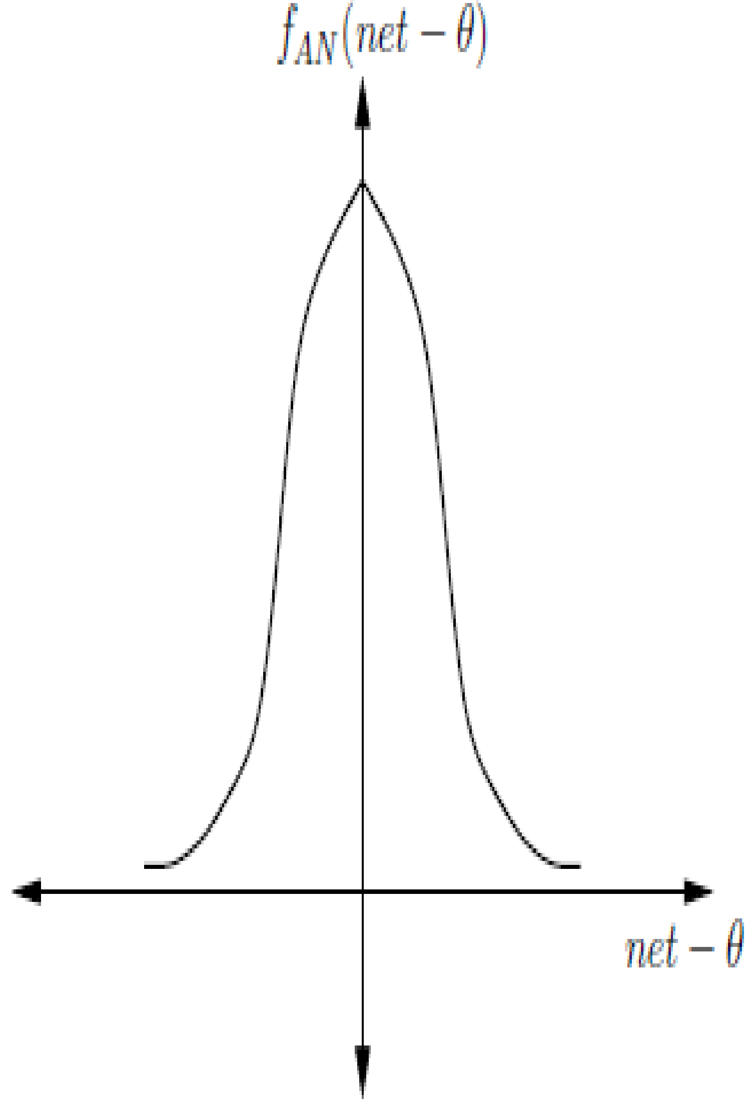
(c) Ramp function



(d) Sigmoid function

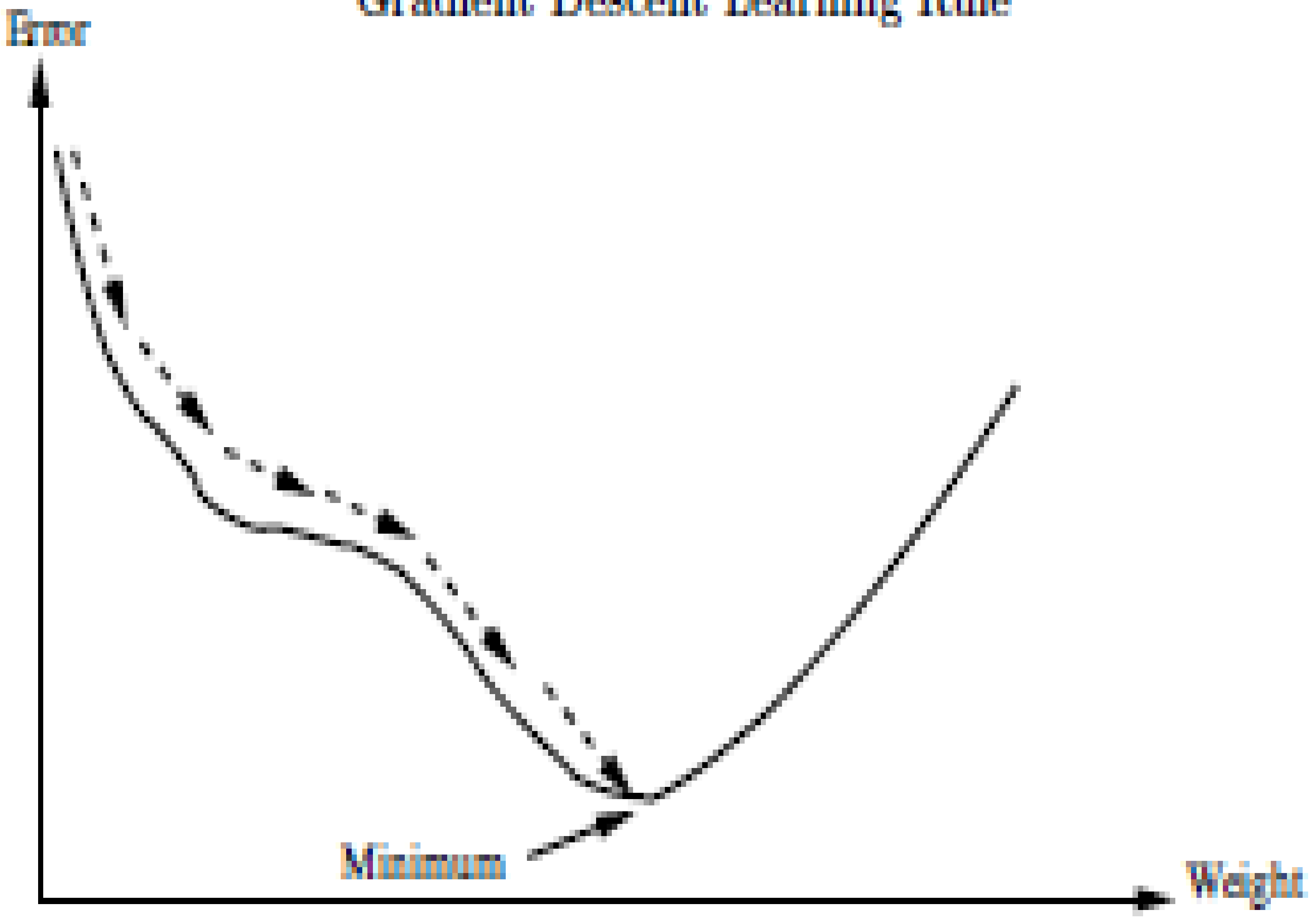


(e) Hyperbolic tangent function

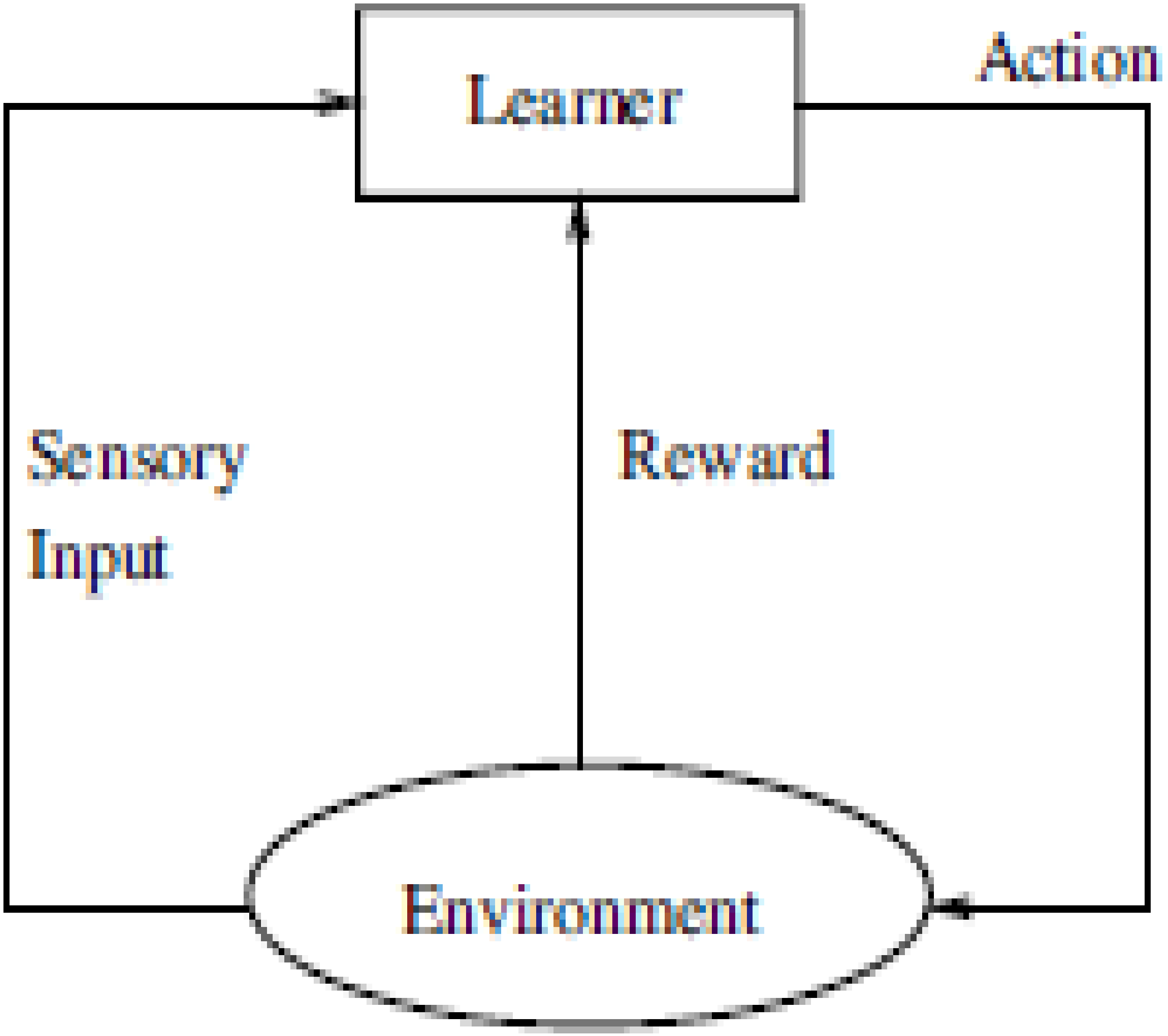


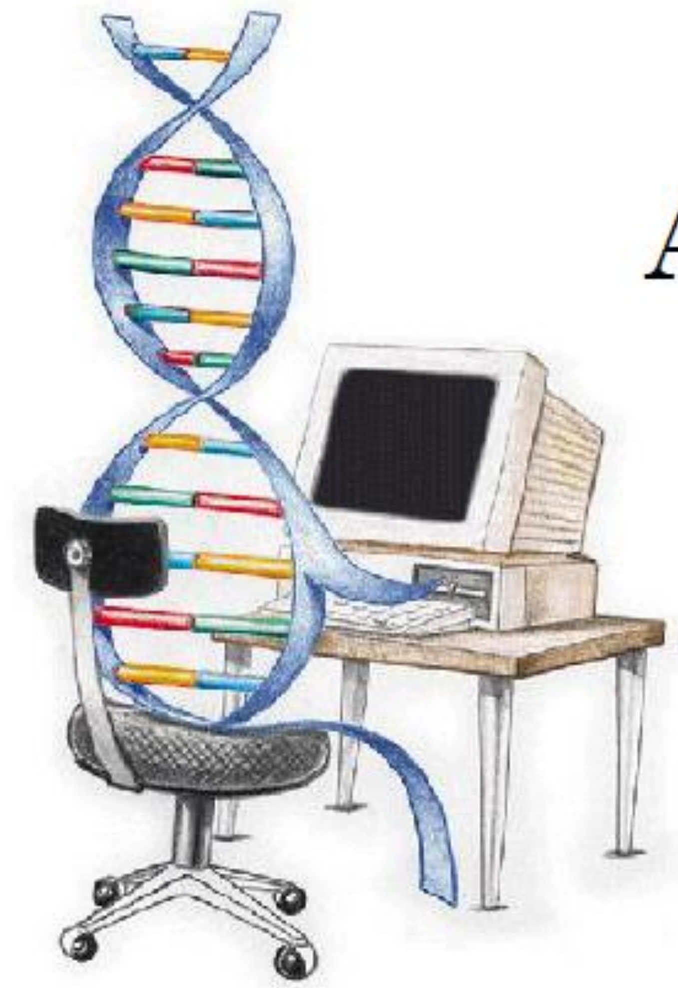
(f) Gaussian function

Gradient Descent Learning Rule



Reinforcement Learning Problem





GENETIC ALGORITHM

WHY GENETIC ALGORITHM?

Genetic Algorithms are good at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, solutions you might not otherwise find in a lifetime.

THE GENETIC ALGORITHM

- Directed search algorithms based on the mechanics of biological evolution.
- Developed by John Holland, University of Michigan (1970's).
- Genetic algorithm provide efficient, effective techniques for optimization and machine learning applications.
- Widely-used today in business, scientific and engineering circles.

TERMINOLOGIES

- Gene–a single encoding of part of the solution space, i.e. either single bits or short blocks of adjacent bits that encode an element of the candidate solution

1

- Chromosome–a string of genes that represents a solution

0	1	0	1	1

- Population–the number of chromosomes available to test

0	1	0	1	1
1	1	1	1	1

1	1	0	1	1
1	0	0	1	1

1	1	0	0	1
0	1	0	1	0

0	1	0	0	0
1	1	0	1	0

EVOLUTIONARY TERMINOLOGY

Abstractions imported from biology

- Chromosomes, Genes, Alleles
- Fitness, Selection
- Crossover, Mutation

THE BASIC GENETIC ALGORITHM

```
{ % Generate random population of chromosomes
    Initialize population;
% Evaluate the fitness of each chromosome in the population
    Evaluate population;[Fitness]
% Create, accept, and test a new population:
    while Termination_Criteria_Not_Satisfied
{% Select according to fitness
    Select parents for reproduction;[Selection]
% With a crossover probability perform crossover or copy parents
    Perform crossover;[Crossover]
% With a mutation probability mutate offspring at each position in chromosome
    Perform mutation;[Mutation]
    Accept new generation;
    Evaluate population;[Fitness]
    }
```

FEATURES OF GENETIC ALGORITHM

- Not too fast but cover large search space
- Capable of quickly finding promising regions of the search space but may take a relatively long time to reach the optimal solution. Solution: hybrid algorithms
- Good heuristics for combinatorial problems
- Usually emphasize combining information from good parents (crossover)
- Different GAs use different
  - –Representations
  - –Mutations
  - –Crossovers
  - –Selection mechanisms

REPRESENTATION

Chromosomes can be:

–Bit strings (0110, 0011, 1101, …)

–Real numbers(33.2, -12.11, 5.32, …)

–Permutations of element(1234, 3241, 4312, …)

–Lists of rules(R1, R2, R3, …Rn…)

–Program elements(genetic programming)

–Any other data structure

CROSSOVER

Crossover is a critical feature of genetic algorithms:

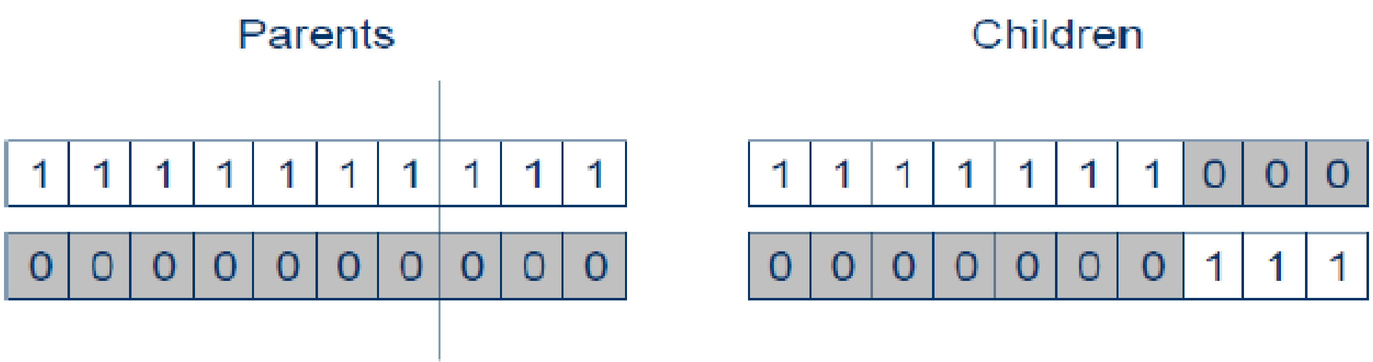
- Some portion of genetic material is swapped between chromosomes
- Typically the swapping produces an offspring
- It greatly accelerates search early in evolution of a population
- It leads to effective combination of schemata (sub solutions on different chromosomes)

Types of Cross Overs:

- 1-Point Crossover
- Two-point crossover
- Uniform crossover

1-POINT CROSSOVER

- Choose a random point
- Split parents at this crossover point
- Create children by exchanging tails
- Probability of crossover is typically in range (0.6, 0.9)



OTHER CROSSOVER TYPES

- Two-point crossover

Parents

1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0

Children

0	0	1	1	1	1	1	0	0	0
1	1	0	0	0	0	0	1	1	1

- Uniform crossover: randomly generated mask

Mask:

1	1	0	1	0	0	0	1	1	0

Parents

1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0

Children

0	0	1	0	1	1	1	0	0	1
1	1	0	1	0	0	0	1	1	0

MUTATION

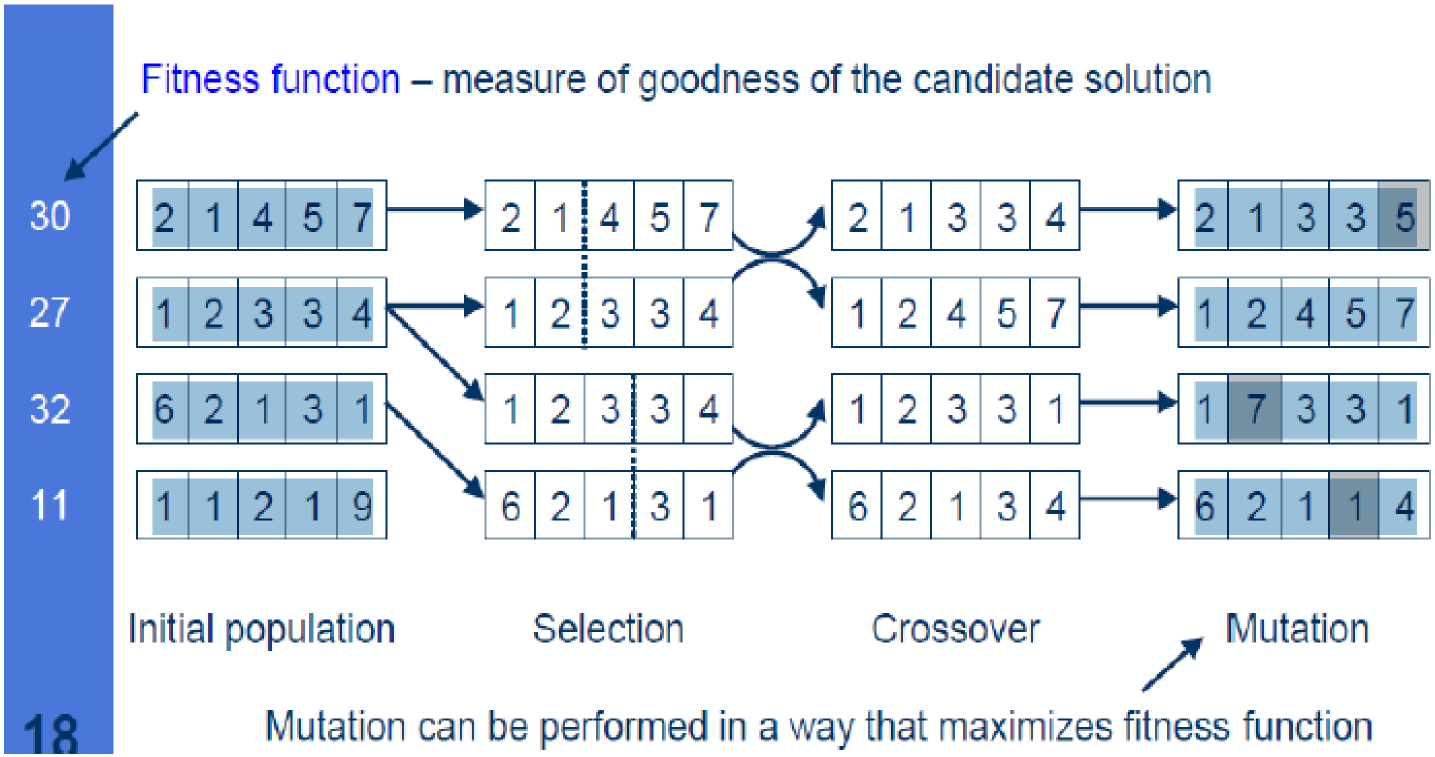
- Alter each gene independently
- Mutation probability is typically in range (1/population_size, 1/chromosome_length)

	Parent		Child
	1 1 1 1 1 1 1 1 1 1		1 1 0 1 1 1 1 0 1 0
	(-4.32 2.12 $\boxed{-41.56}$ 9.99)		(-4.32 2.12 $\boxed{-43.15}$ 9.99)

- Choose mutation with the best fit

SELECTION

- Parents with better fitness have better chances to produce offspring

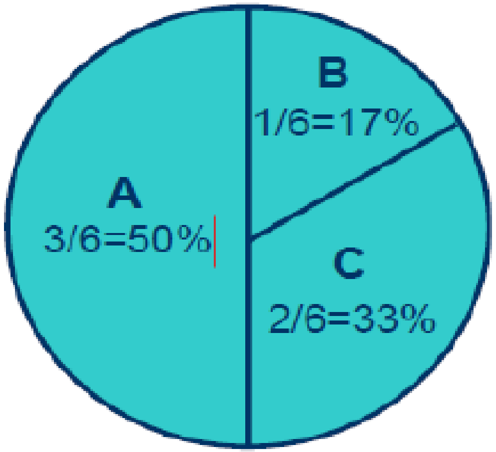


SELECTION: ROULETTE WHEEL

- Better solutions get higher chance to become parents for next generation solutions
- Roulette wheel technique:

  –Assign each individual part of the wheel

  –Spin wheel N times to select N individuals



fitness(A) = 3

fitness(B) = 1

fitness(C) = 2

BENEFITS OF GENETIC ALGORITHMS

- Concept is easy to understand.
- Modular–separate from application (representation); building blocks can be used in hybrid applications.
- Supports multi-objective optimization.
- Good for "noisy"environment.
- Always results in an answer, which becomes better and better with time.
- Can easily run in parallel.
- The fitness function can be changed from iteration to iteration, which allows incorporating new data in the model if it becomes available.

ISSUES WITH GENETIC ALGORITHMS

- Choosing parameters:
  - –Population size
  - –Crossover and mutation probabilities
  - –Selection, deletion policies
  - –Crossover, mutation operators, etc.
  - –Termination criteria
- Performance:
  - –Can be too slow but covers a large search space
  - –Is only as good as the fitness function

ISSUES FOR GA PRACTITIONERS

- Choosing basic implementation issues:
  - representation
  - population size, mutation rate, ...
  - selection, deletion policies
  - crossover, mutation operators
- Termination Criteria
- Performance, scalability
- Solution is only as good as the evaluation function (often hardest part)

APPLICATIONS OF GENETIC ALGORITHMS

- ***Optimization***–numerical and combinatorial optimization problems, e.g. traveling salesman, routing, graph colouringand partitioning
- ***Robotics***–trajectory planning
- ***Machine learning***–designing neural networks, classification and prediction, e.g. prediction of weather or protein structure,
- ***Signal processing***–filter design
- ***Design***–semiconductor layout, aircraft design, communication networks
- ***Automatic programming***–evolve computer programs for specific tasks, design cellular automata and sorting networks
- ***Economics***–development of bidding strategies, emergence of economics markets
- ***Immune systems***–model somatic mutations
- ***Ecology***–model symbiosis, resource flow
- ***Population genetics***–"Under what condition will a gene for recombination be evolutionarily viable?"

SOME GA APPLICATION TYPES

Domain	Application Types
Control	gas pipeline, pole balancing, missile evasion, pursuit
Design	semiconductor layout, aircraft design, keyboard configuration, communication networks
Scheduling	manufacturing, facility scheduling, resource allocation
Robotics	trajectory planning
Machine Learning	designing neural networks, improving classification algorithms, classifier systems
Signal Processing	filter design
Game Playing	poker, checkers, prisoner's dilemma
Combinatorial Optimization	set covering, travelling salesman, routing, bin packing, graph colouring and partitioning

FUZZY LOGIC & CLASSIFICATION AN OVERVIEW

INTRODUCTION

- Fuzzy Logic was initiated in 1965, by Dr. Lotfi A. Zadeh, professor for computer science at the university of California in Berkley.

- Basically, Fuzzy Logic is a multivalued logic, that allows intermediate values to be defined between conventional evaluations like true/false, yes/no, high/low, etc.

- Fuzzy Logic starts with and builds on a set of user–supplied human language rules.

- Fuzzy Systems convert these rules to their mathematical equivalents.

- This simplifies the job of the system designer and the computer, and results in much more accurate representations of the way system behaves in real world.

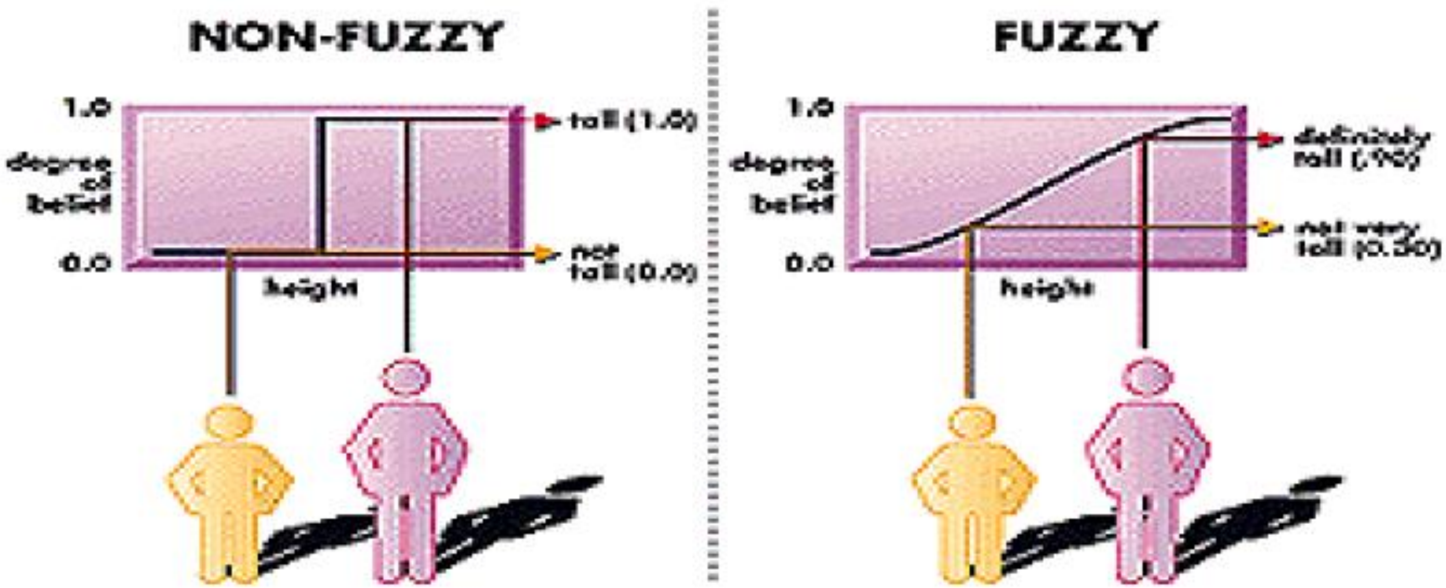
- Fuzzy Logic provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information.

9/6/

FUZZY LOGIC

➤ What is Fuzzy Logic?

Fuzzy Logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth, i.e. truth values between "completely true" and "completely false".



FUZZY LOGIC

➢ How Fuzzy Logic works?

- In Fuzzy Logic, unlike standard conditional logic, the truth of any statement is a matter of degree. (e.g How cold is it? How high shall we set the heat?)

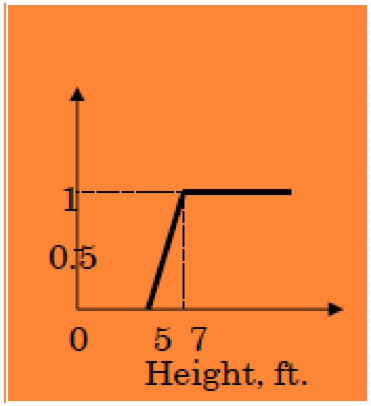
- The degree to which any Fuzzy statement is true is denoted by a value between 0 and 1.

- Fuzzy Logic needs to be able to manipulate degrees of "may be" in addition to true and false.

➢ Example:

tall(x) = {
    0 , if height(x) < 5 ft.,
    (height(x)-5ft.)/2ft. , if 5 ft. <= height (x) <= 7 ft.,
    1 , if height(x) > 7 ft.
}

U: universe of discourse (i.e. set of people)
TALL: Fuzzy Subset



FUZZY LOGIC (CONTD.)

- Given the above definitions, here are some example values.

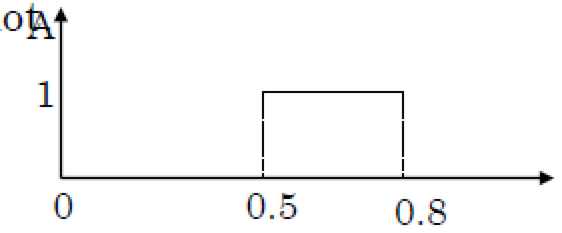
Person	Height	degree of tallness
Billy	3' 2"	0.00
Yoke	5' 5"	0.21
Drew	5' 9"	0.38
Erik	5' 10"	0.42
Mark	6' 1"	0.54
Kareem	7' 2"	1.00

- From this definitions, we can say that,

  - the degree of truth of the statement “Drew is TALL” is 0.38.

FUZZY SETS

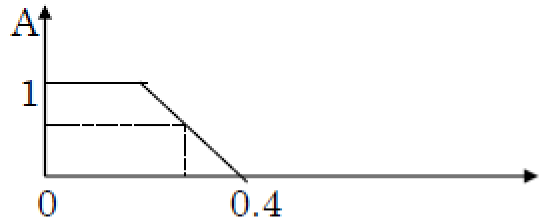
- In classical mathematics we are familiar with what we call *crisp* sets. In this method, the characteristic function assigns a number 1 or 0 to each element in the set, depending on whether the element is in the subset A or not.



1→ In set A

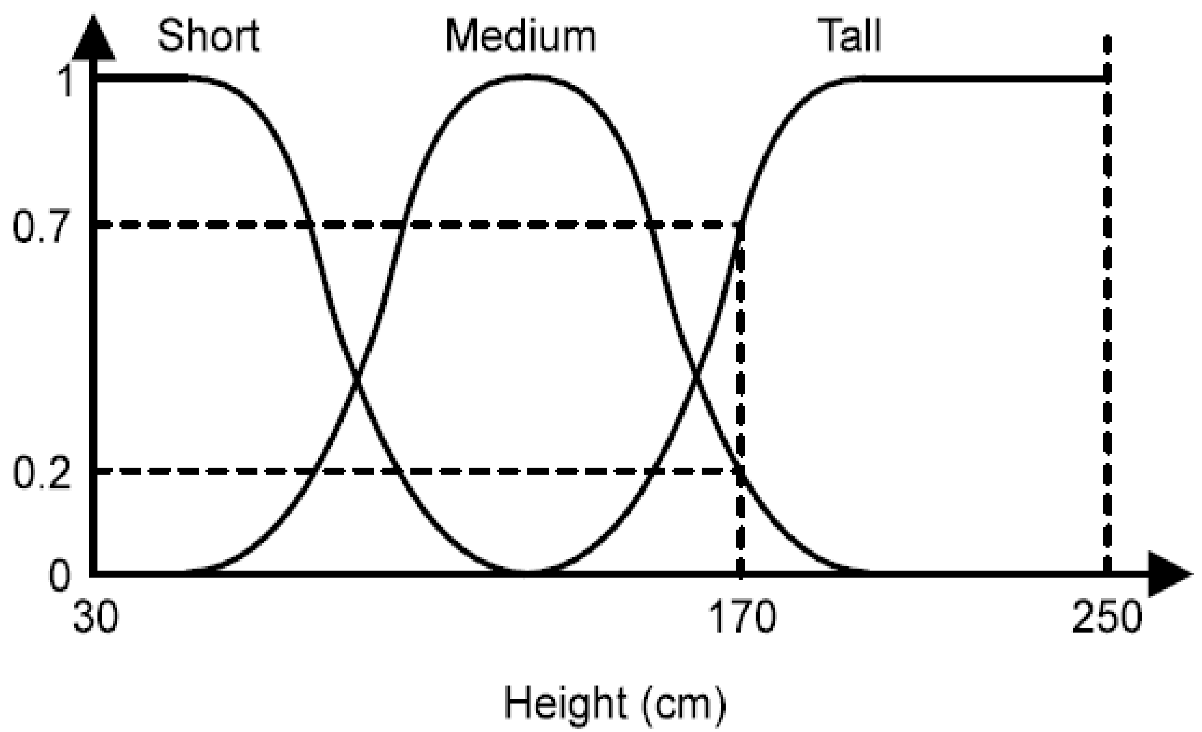
0→ Not in set A

- This concept is sufficient for many areas of application, but it lacks flexibility for some applications like classification of remotely sensed data analysis.
- The membership function is a graphical representation of the magnitude of participation of each input. It associates weighting with each of the inputs that are processed.



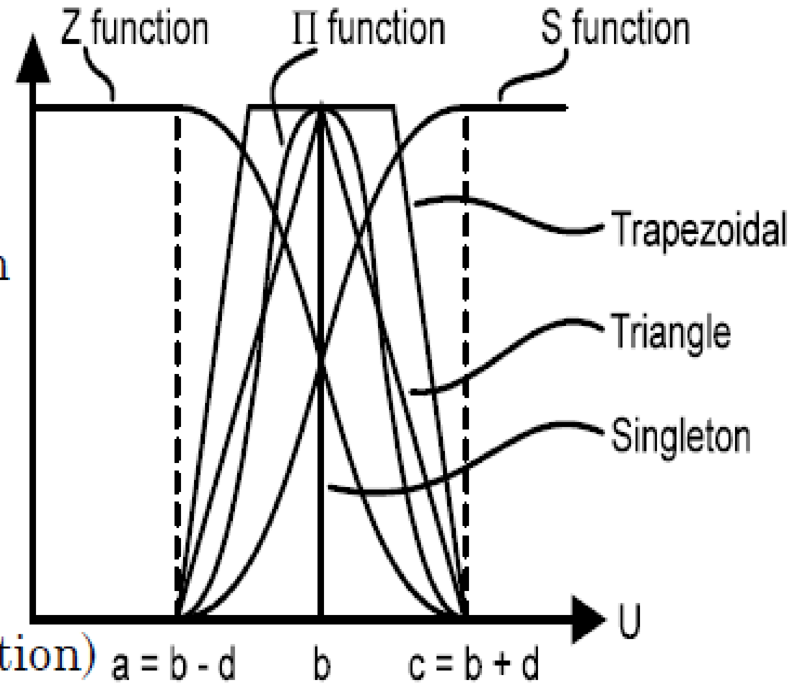
FUZZY SETS (CONTD.)

➢ Membership Functions representing three fuzzy sets for the variable "height".



FUZZY SETS (CONTD.)

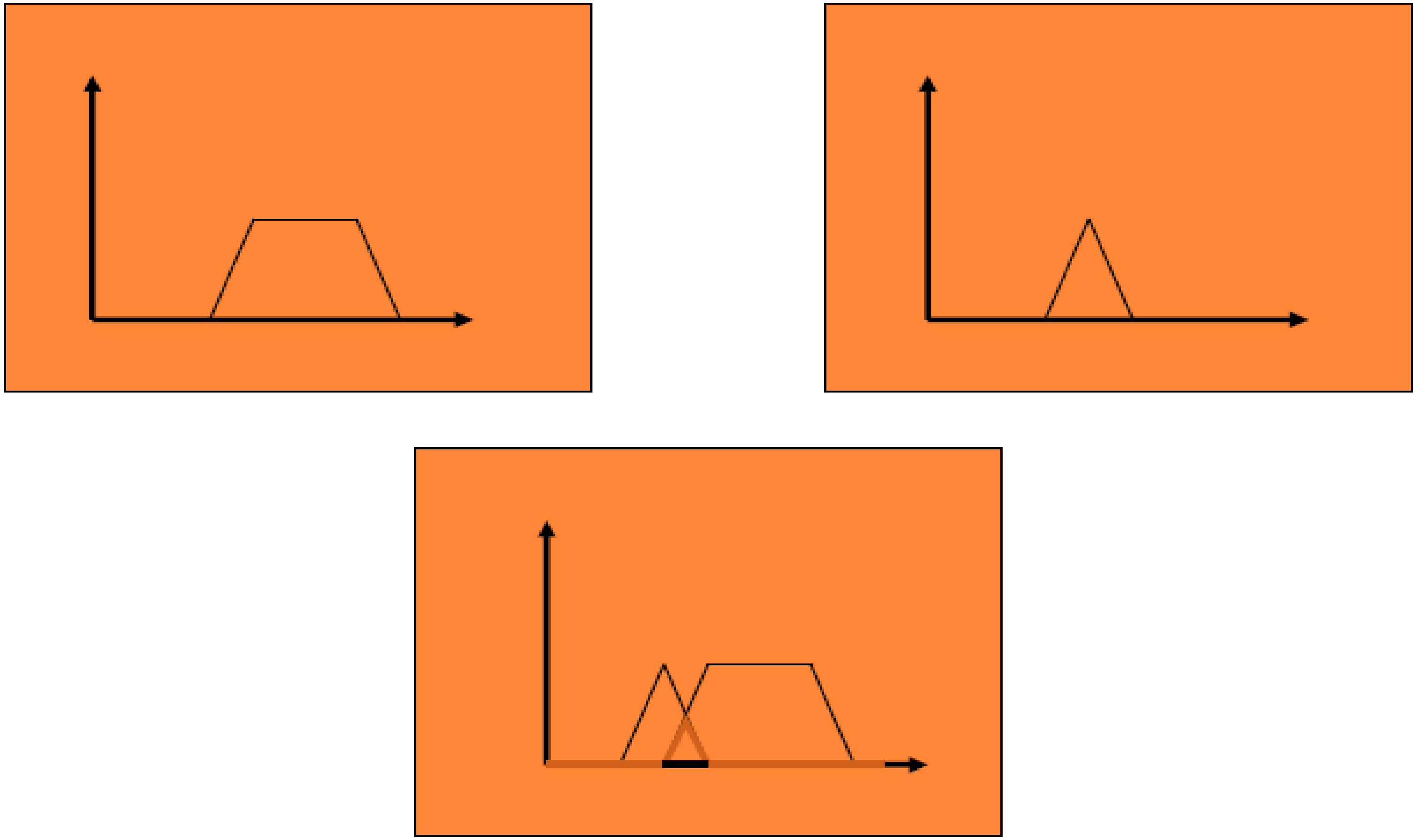
- Standard Membership Functions:
  - Single-Valued, or Singleton
  - Triangular
  - Trapezoidal
  - S- Function (Sigmoid Function)



Different types of Membership Functions.

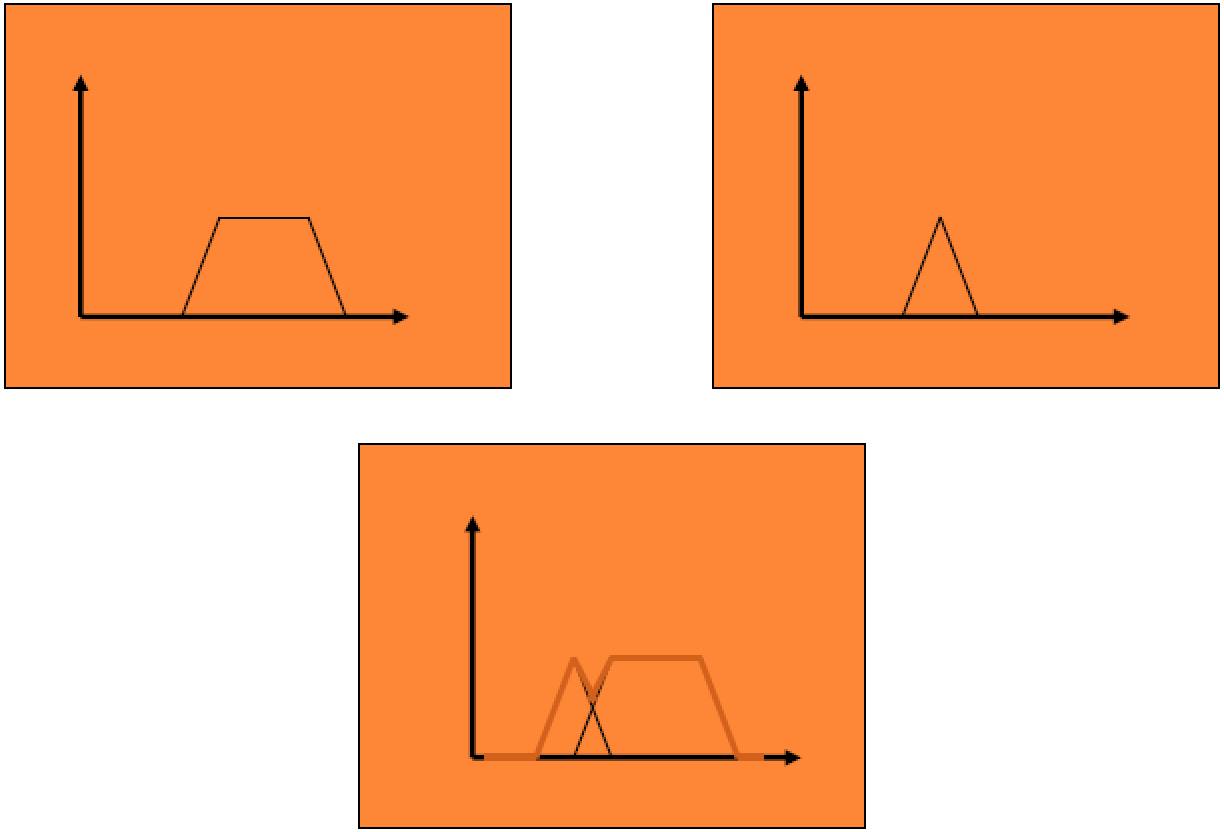
OPERATIONS ON FUZZY SETS

Fuzzy *AND*:



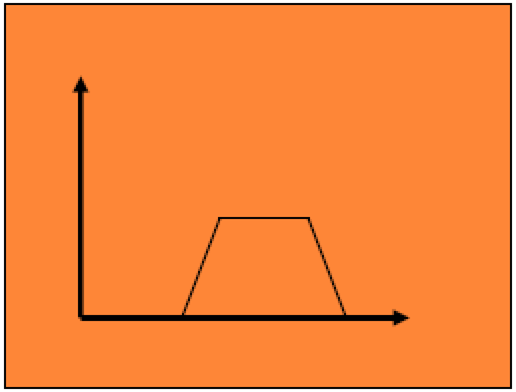
OPERATIONS ON FUZZY SETS (CONTD.)

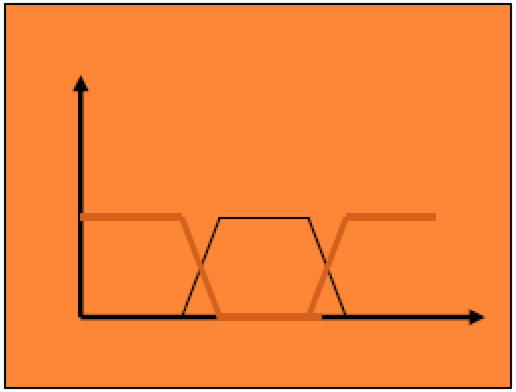
Fuzzy *OR*:



OPERATIONS ON FUZZY SETS (CONTD.)

Fuzzy *NOT*:





PROPERTIES

The following rules which are common in classical set theory also apply to Fuzzy Logic.

- De Morgan's $\overline{(A \cap B)} = \overline{A} \cap \overline{B}$ $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$
- Associativity:

$$(A \cap B) \cap C = A \cap (B \cap C)$$
$$(A \cup B) \cup C = A \cup (B \cup C)$$

- Commutativity:

$$A \cap B = B \cap A \text{ , } A \cup B = B \cup A$$

- Distributivity:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$
$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

PROBABILITY VS FUZZY LOGIC

Probability	Fuzzy Logic
Probability Measure	Membership Function
Before an event happens	After it happened
Measure Theory	Set Theory

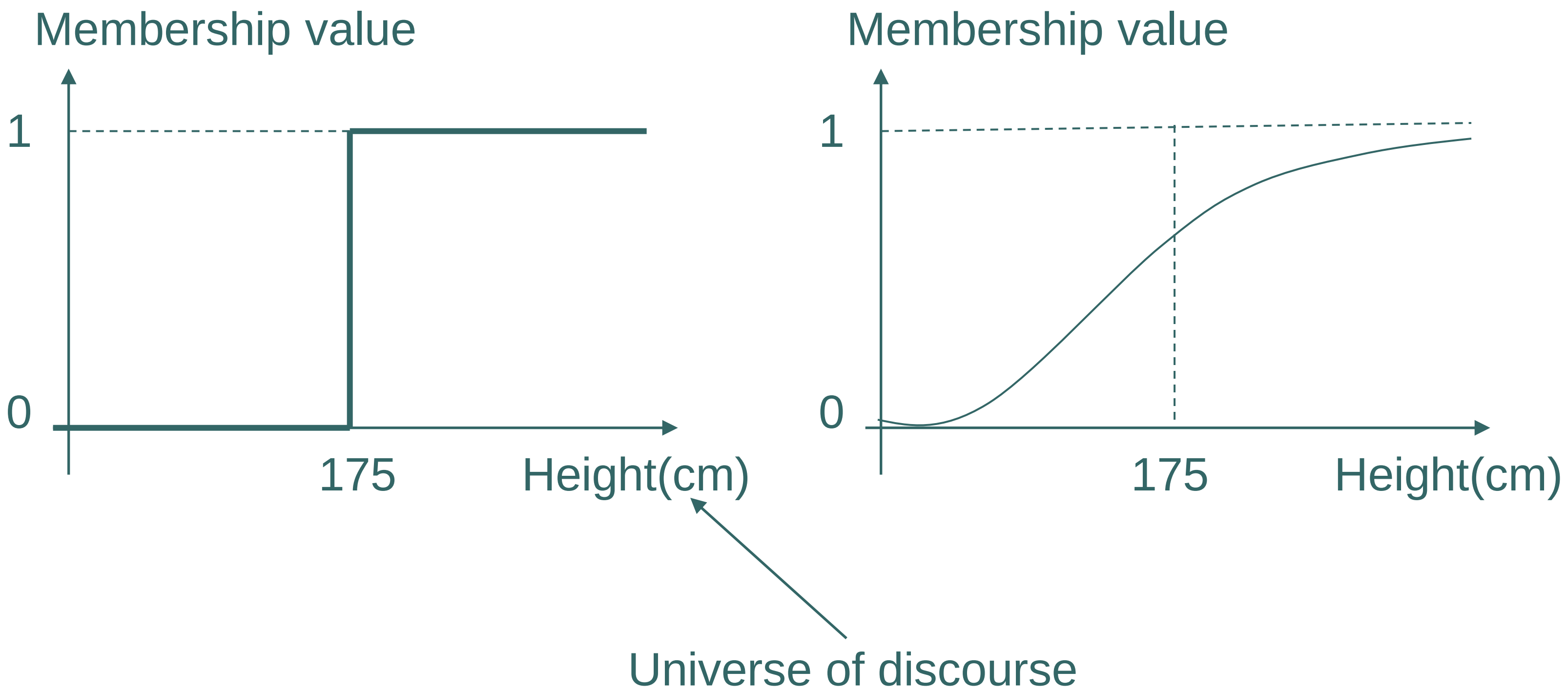
FUZZY SYSTEMS

Fuzzy Sets Theory

- Boolean logic
  - Uses sharp distinctions. It forces us to draw a line between a members of class and non members.
- Fuzzy logic
  - Reflects how people think. It attempt to model our senses of words, our decision making and our common sense -> more human and intelligent systems

Fuzzy Sets Theory

- Classical Set vs Fuzzy set



Fuzzy Sets Theory

- Classical Set vs Fuzzy set

Let X be the universe of discourse and its elements be denoted as x.
In the classical set theory, crisp set A of X is defined as function $f_A(x)$ called the the characteristic function of A

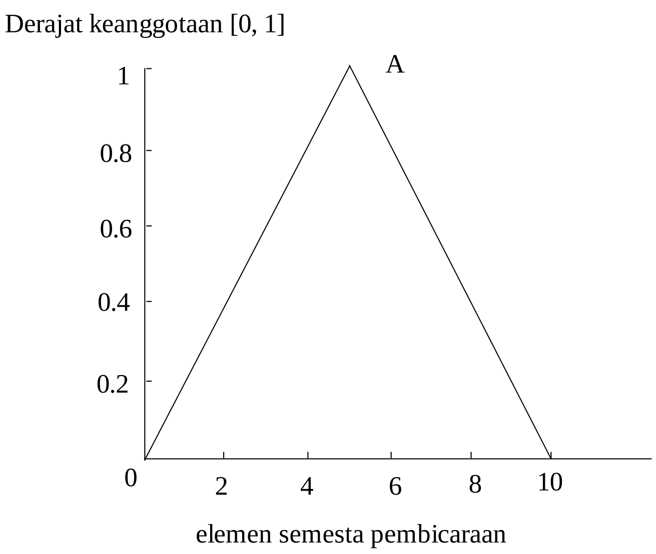
$$f_A(x) : X \to \{0,1\}, \text{ where } f_A(x) = \begin{cases} 1, \text{if } x \in A \\ 0, \text{if } x \notin A \end{cases}$$

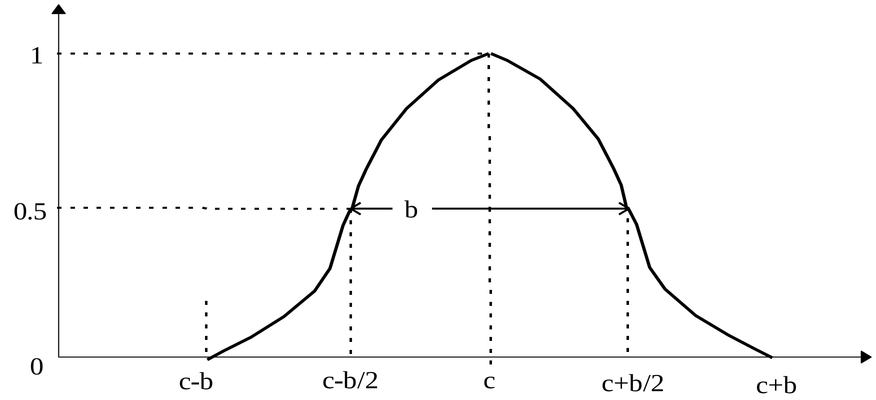
In the fuzzy theory, fuzzy set A of universe of discourse X is defined by function $\mu_A(x)$ called the membership function of set A

$$\mu_A(x) : X \to [0,1], \mathit{where}\ \mu_A(x) = 1 \text{ if } x \text{ is totally in } A;$$
$$\mu_A(x) = 0 \text{ if } x \text{ is not in } A;$$
$$0 < \mu_A(x) < 1 \text{ if } x \text{ is partly in } A.$$

Fuzzy Sets Theory

- Membership function



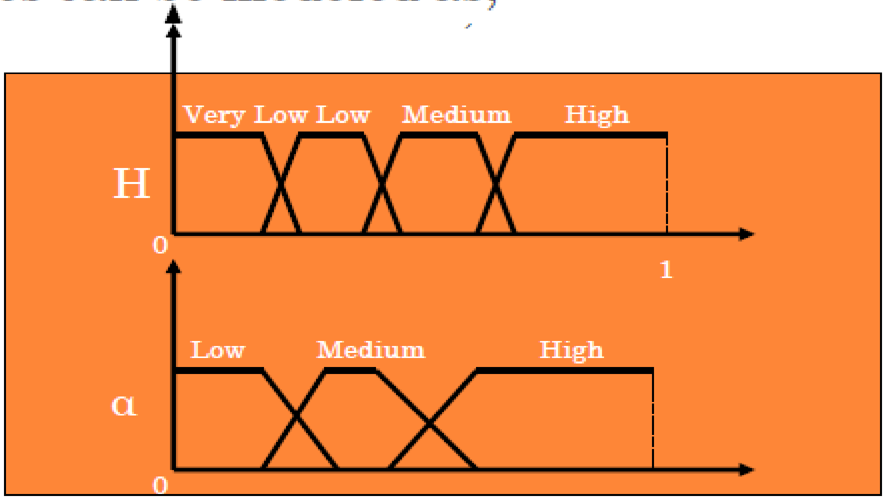


FUZZY CLASSIFICATION

- Fuzzy classifiers are one application of fuzzy theory.

- *Expert knowledge* is used and can be expressed in a very natural way using linguistic variables, which are described by fuzzy sets.

- For Example: consider two variables Entropy H and α-angle. These variables can be modeled as;



FUZZY CLASSIFICATION (CONTD.)

- In fuzzy classification, a sample can have membership in many different classes to different degrees. Typically, the membership values are constrained so that all of the membership values for a particular sample sum to 1.

- Now the *expert knowledge* for this variable can be formulated as a rule like

IF Entropy *high* AND α *high* THEN Class = class 4

- The rules can be combined in a table, called as rule base.

FUZZY CLASSIFICATION (CONTD.)

Entropy	α	*Class*
Very low	Low	Class 1
Low	Medium	Class 2
Medium	High	Class 3
High	High	Class 4

Example for a fuzzy rule base

FUZZY CLASSIFICATION (CONTD.)

- Linguistic rules describing the control system consist of two parts; an antecedent block (between the IF and THEN) and a consequent block (following THEN).

- Depending on the system, it may not be necessary to evaluate every possible input combination, since some may rarely or never occur.

- Optimum evaluation is usually done by experienced operators.

- The inputs are combined logically using the AND operator to produce output response values for all expected inputs. The active conclusions are then combined to logical sum for each membership function.

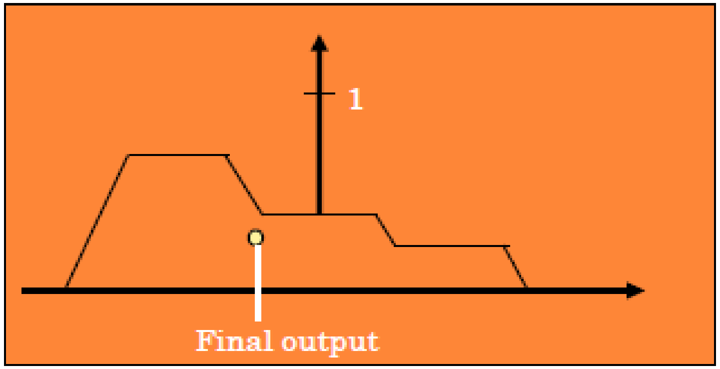
- Finally, all that remains is combined in defuzzyfication process to produce the crisp output.

FUZZY CLASSIFICATION (CONTD.)

- To obtain a crisp decision from this fuzzy output, we have to defuzzify the fuzzy set. Therefore, we have to choose one representative value.

- There are several methods of defuzzification, one of them is to take the center of gravity of the fuzzy set. This is a widely used method for fuzzy sets.

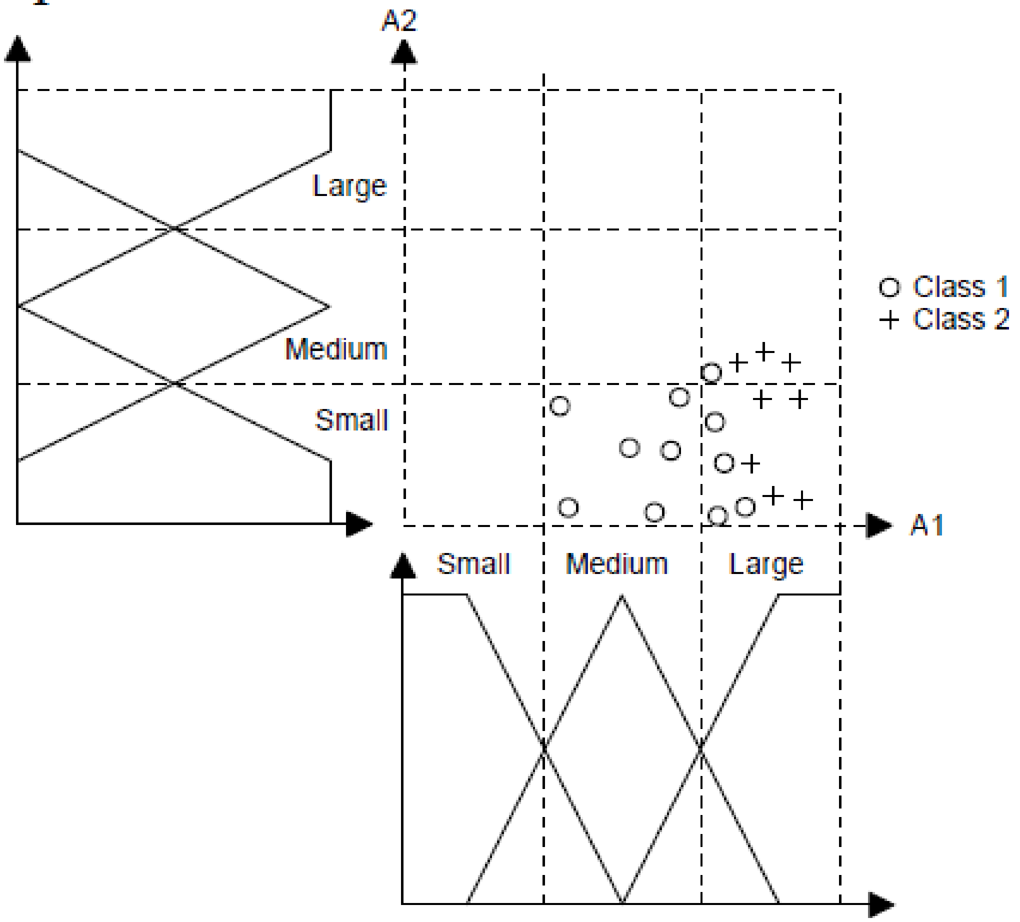
- For Example:



Defuzzification using the center of gravity approach

FUZZY CLASSIFICATION (CONTD.)

- Another Example:



PROS & CONS

- Advantages:

  - Helpful for very complex or highly nonlinear processes.
  - Allows use of "fuzzy" concepts like medium, low, etc.
  - Biggest impact is for control problems.
  - Help avoid discontinuities in behavior.

- Disadvantages:

  - Sometimes results are unexpected and hard to debug.
  - Computationally complicated.
  - According to literature, Fuzzy Logic is not recommendable, if conventional approach yields a satisfying result.

CONCLUSION

- **Soft Computing** differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximation.
- In effect, the role model for **soft computing** is the human mind. Principal constituents of **Soft Computing** are Neural Networks, Fuzzy Logic, Evolutionary Computation, Swarm Intelligence and Bayesian Networks.
- The successful applications of soft computing suggest that the impact of **soft computing** will be felt increasingly in coming years.
- Soft computing plays an important role in science and engineering, but eventually its influence may extend much farther.

Applications of Computational Intelligence Techniques in Engineering

- **Computer Science**
- **Engineering**
- **Data Analysis**
- **Bio - Medicine**

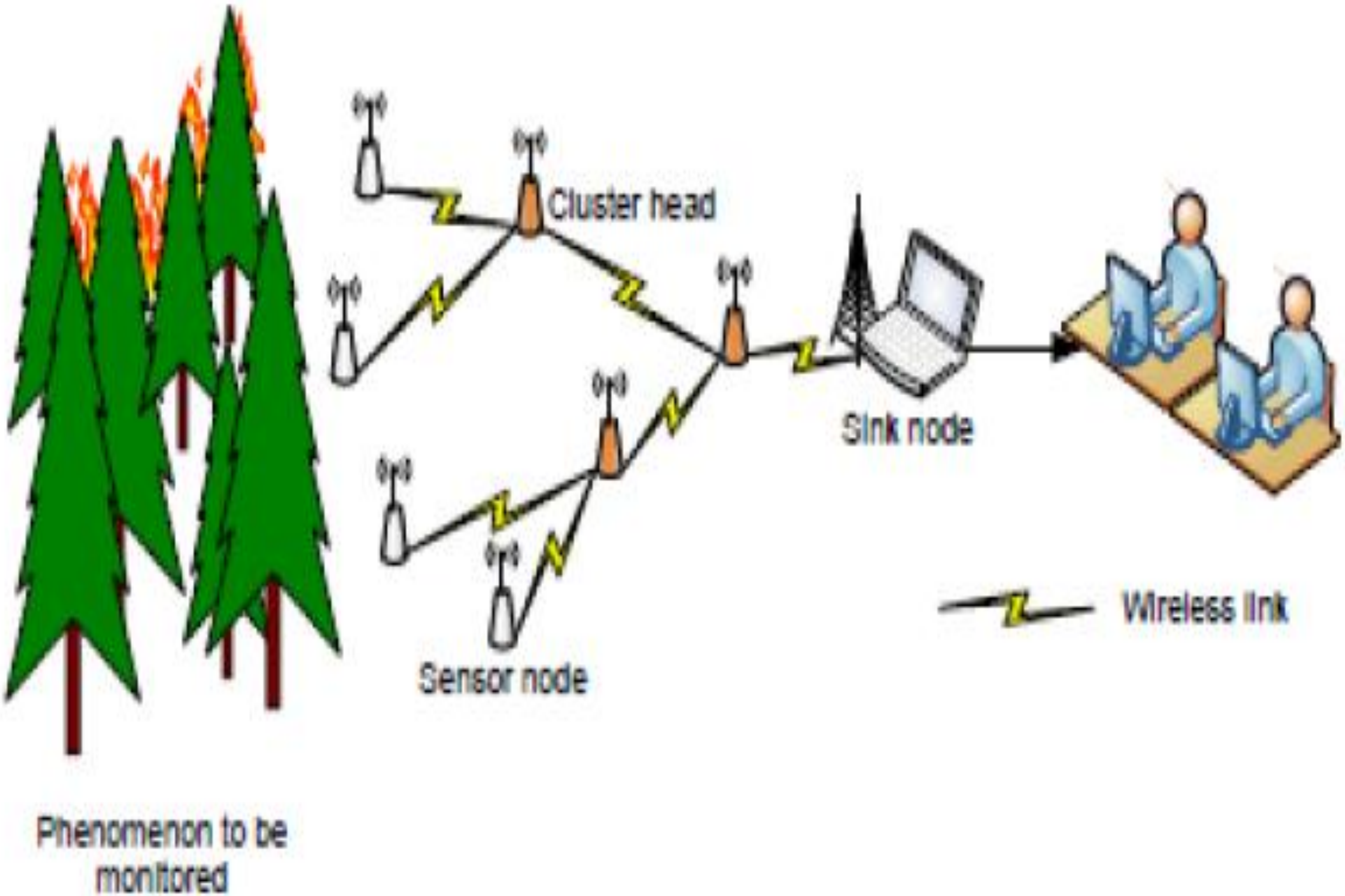
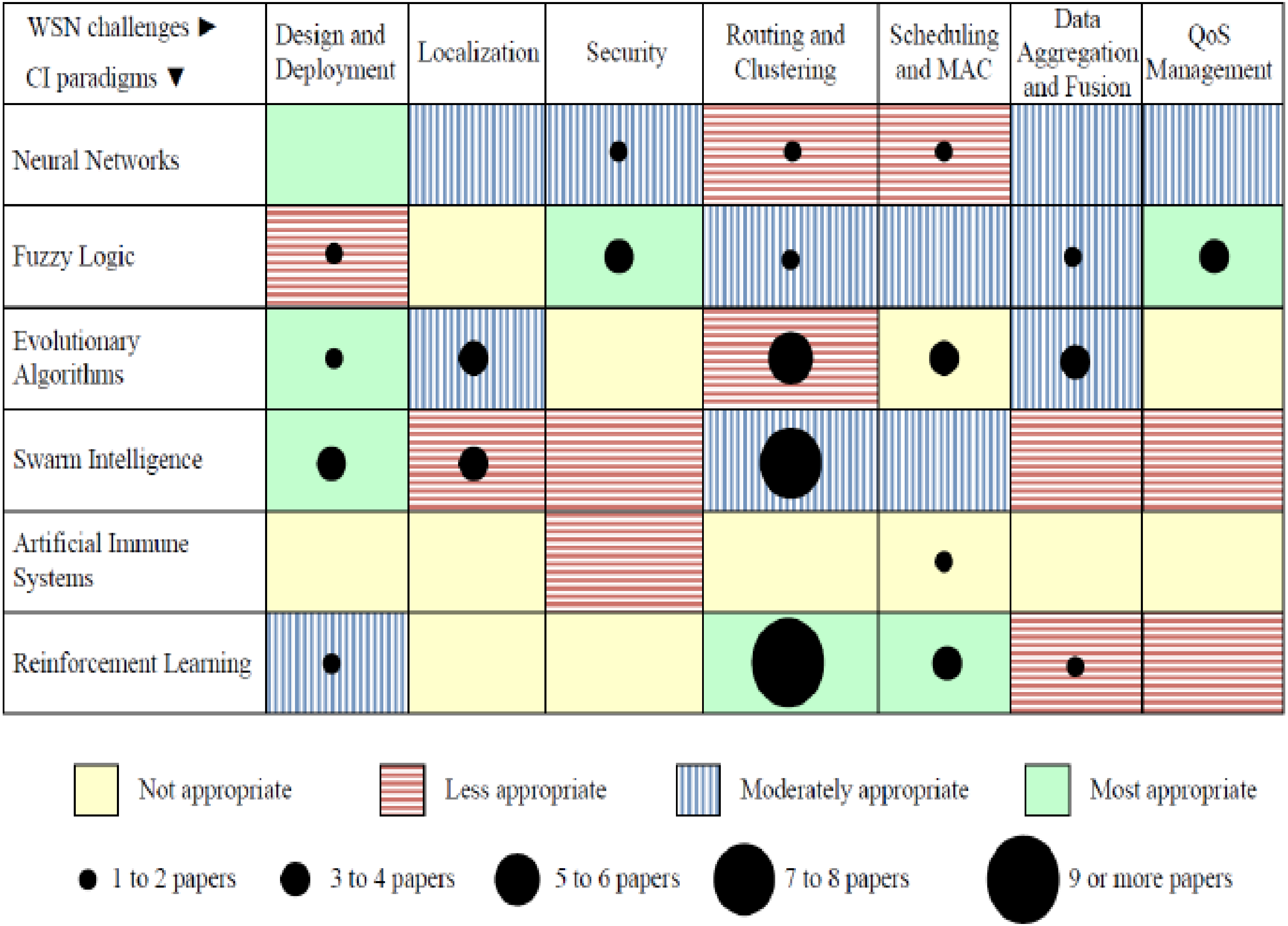


Figure 1.1 Architecture of a Typical WSN


WSN challenges ▶
CI paradigms ▼
Design and Deployment
Localization
Security
Routing and Clustering
Scheduling and MAC
Data Aggregation and Fusion
QoS Management
Neural Networks
Fuzzy Logic
Evolutionary Algorithms
Swarm Intelligence
Artificial Immune Systems
Reinforcement Learning
Not appropriate
Less appropriate
Moderately appropriate
Most appropriate
1 to 2 papers
3 to 4 papers
5 to 6 papers
7 to 8 papers
9 or more papers

Thank You