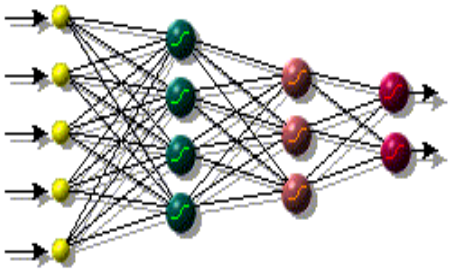
ARTIFICIAL NEURAL NETWORKS



Dr. Parimal Acharjee
Professor
Electrical Engineering
National Institute of Technology Durgapur

Biological Inspiration

- Inspired by biological nervous systems
- The information processing cells of the brain are the neurons.
- Electro-chemical signals are propagated
- Parallel computing
- Powerful predictive tool for large no of variables
- Central theme of artificial neural network (ANN) is borrowed from biological neural network (BNN)

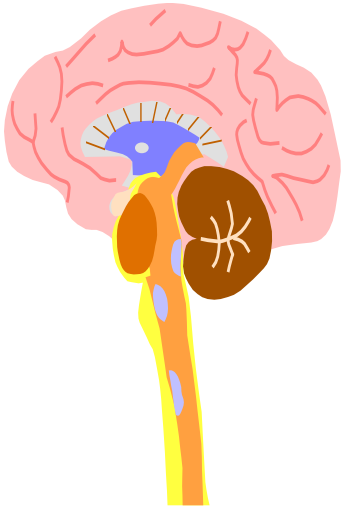
Characteristics of Biological Neural Networks (BNN)

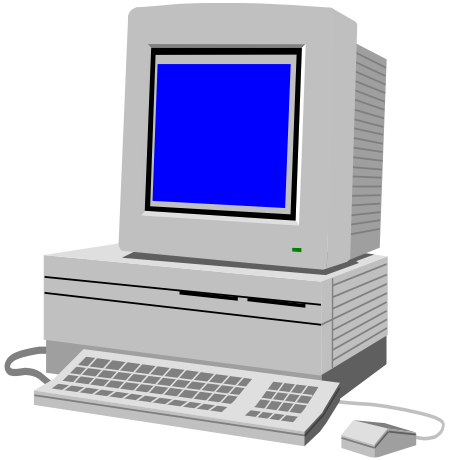
1) Massive connectivity
2) Nonlinear, Parallel, Robust and Fault Tolerant
3) Capability to adapt to surroundings
4) Ability to learn and generalize from known examples
5) Collective behavior is different from individual behavior

Artificial Neural Networks mimics some of the properties of the biological neural networks

Brain and Machine

- The Brain
  - Pattern Recognition
  - Association
  - Complexity
  - Noise Tolerance





- The Machine
  - Calculation
  - Precision
  - Logic

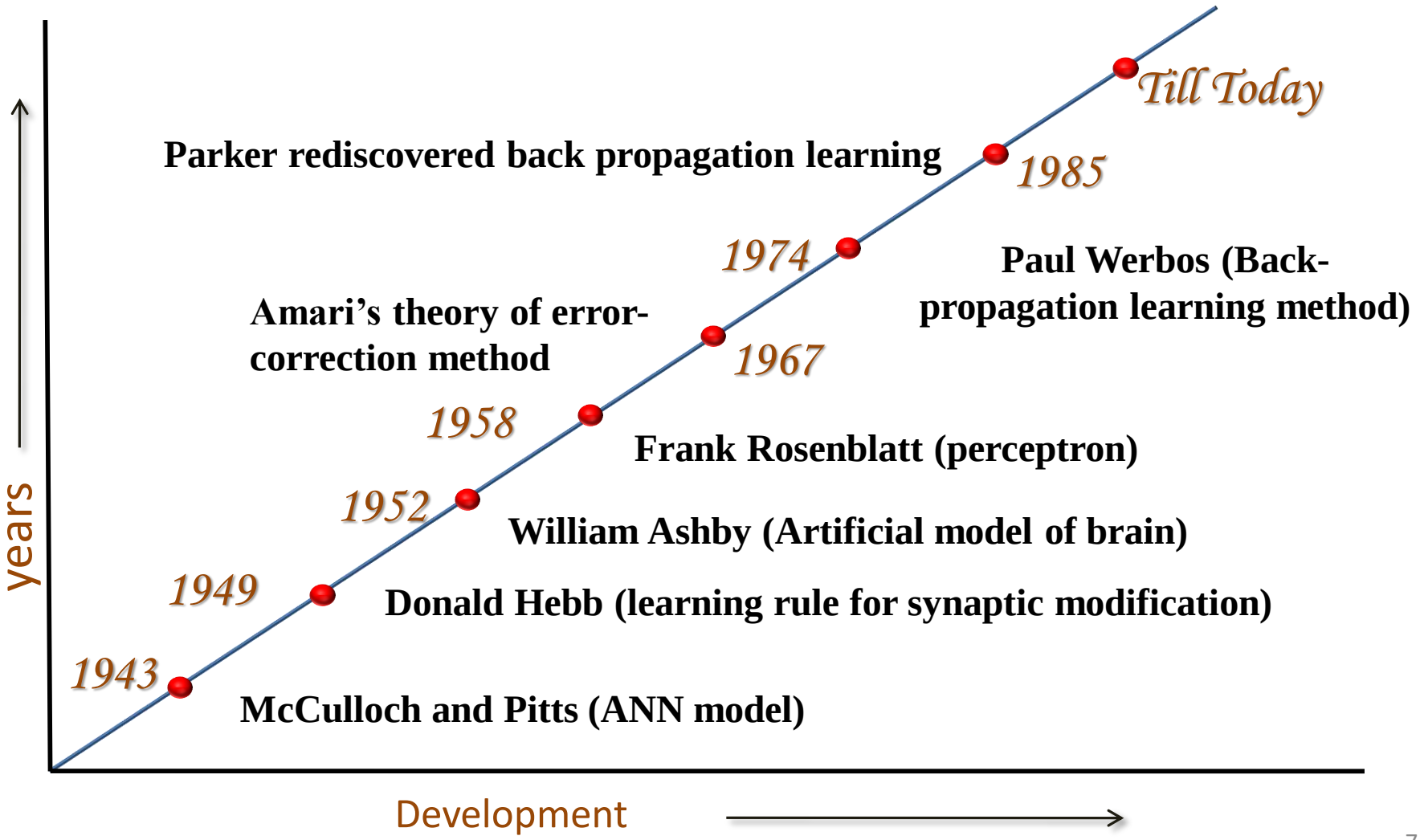
Features of the Brain

- Highly complex, nonlinear and parallel processing system
- Ten billion (10^{10}) neurons
- 60 trillion connections or synapses
- Neuron switching time $>10^{-3}$secs
- Face Recognition ~0.1secs
- On average, each neuron has several thousand connections
- Hundreds of operations per second
- High degree of parallel computation
- Distributed representations

Definition

- A **Neural Network** is a network of some processing elements called neurons, that can be used to determine input-output relationship of a complex process. It may be considered as a non-linear statistical data-modelling process.
- **Artificial Neural Networks** are massively parallel adaptive networks of simple nonlinear computing elements called neurons which are intended to abstract and model some of the functionality of the human nervous system in an attempt to partially capture some of its computational strength.

Historical Background



Neurons or neurotransmitters

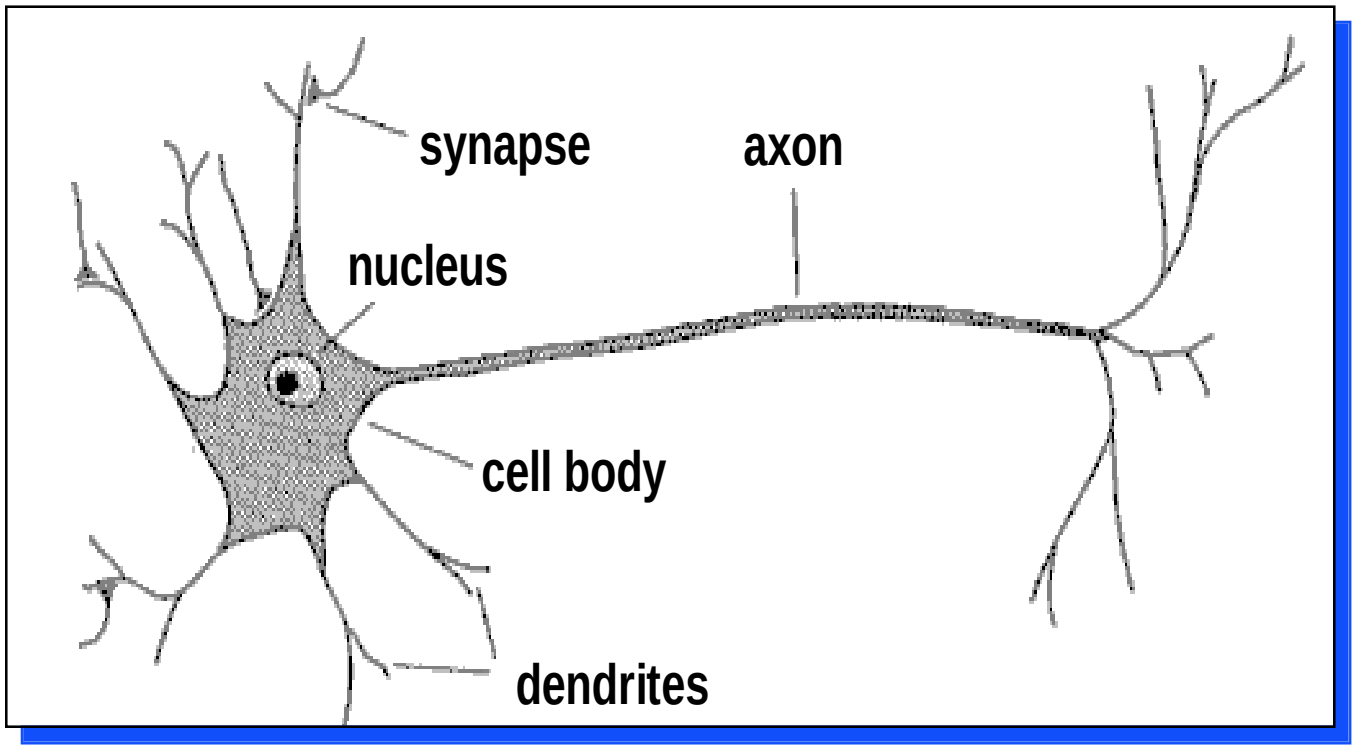
- Neurons or neurotransmitters are nerve cells that process and transmit information through electrical and chemical signals.
- There are several types of neurons; one type of which are sensory neurons which respond to touch, light, sound, and other stimuli and send signals to the spinal cord and the brain. Motor neurons then receive signals from the brain and spinal cord and cause muscles to contract and affect the glands. They connect to each other and form networks and communicate through synapses which are contained in the brain.
- Synapses are junctions that allow a neuron to electrically or chemically transmit a signal to another cell.

The Structure of Neurons

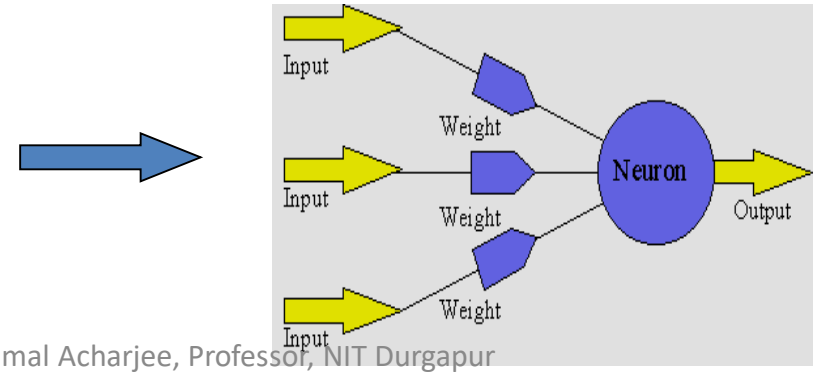
A neuron has a cell body, a branching input structure (the dendrites) and a branching output structure (the axon).

- The cell itself includes a necleus (at the center)
- The dendrites provide input signal to the cell
- Axons connect to dendrites via synapses and send output signal.
- Electro-chemical signals are propagated from the dendritic input, through the cell body, and down the axon to other neurons

The Structure of Neurons



Analogous ANN



The Structure of Neurons

- A neuron only fires if its input signal exceeds a certain amount (the threshold) in a short time period.
- Synapses vary in strength
  - Good connections allowing a large signal
  - Slight connections allow only a weak signal.
  - Synapses can be either excitatory or inhibitory.

Inhibitory vs Excitatory

- Ever wonder why we act and react differently to various stimuli?
- The human body is composed of various elements that react differently to various stimuli through the nervous system.
- Synapses can either be excitatory or inhibitory. Inhibitory synapses decrease the likelihood of the firing action potential of a cell while excitatory synapses increase its likelihood. Excitatory synapses cause a positive action potential in neurons and cells.
- Excitatory synapses polarize neurotransmitters in the postsynaptic membrane while inhibitory synapses depolarize them.
- Excitatory synapses stimulate neurotransmitters while inhibitory synapses inhibit them.

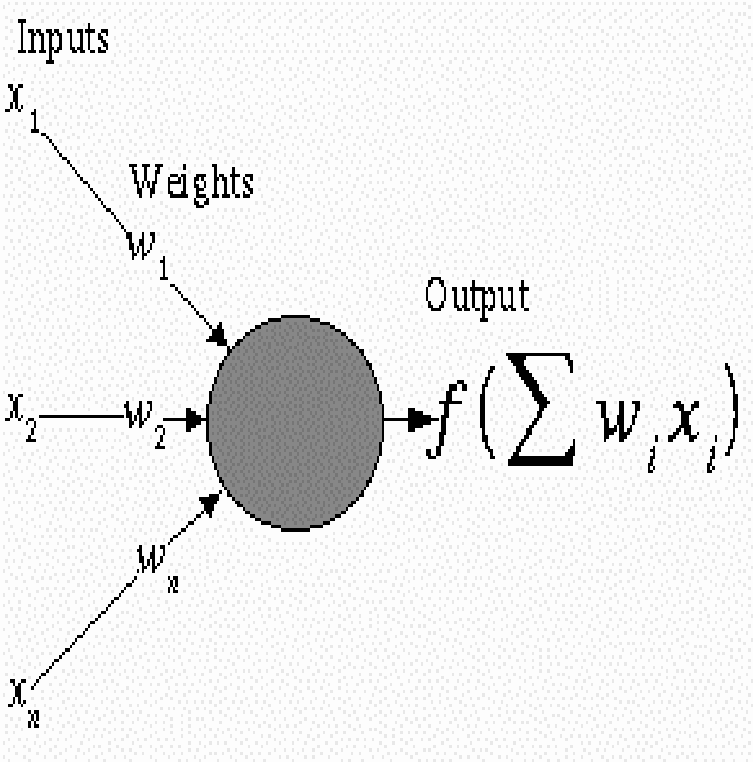
NEURAL NETWORK REPRESENTATION

- An ANN is composed of processing elements called ***perceptrons***, organized in different ways to form the network's structure.

Processing Elements

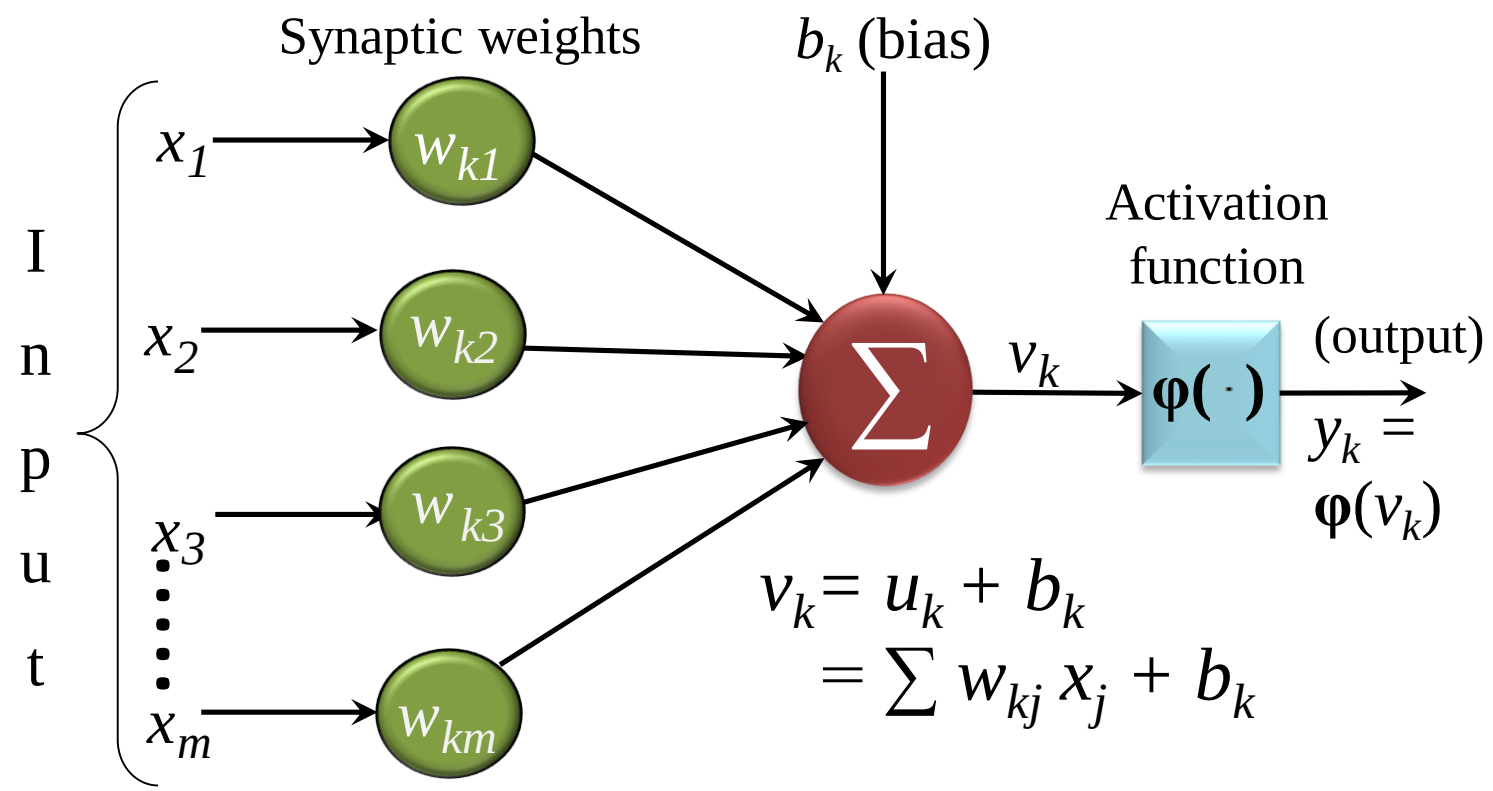
- An ANN consists of perceptrons. Each of the perceptrons receives inputs, processes inputs and delivers a single output.

The input can be raw input data or the output of other perceptrons. The output can be the final result (e.g. 1 means yes, 0 means no) or it can be inputs to other perceptrons.



Model of an Artificial Neuron (Perceptron)

A perceptron (P) calculates the weighted sum of the input values.



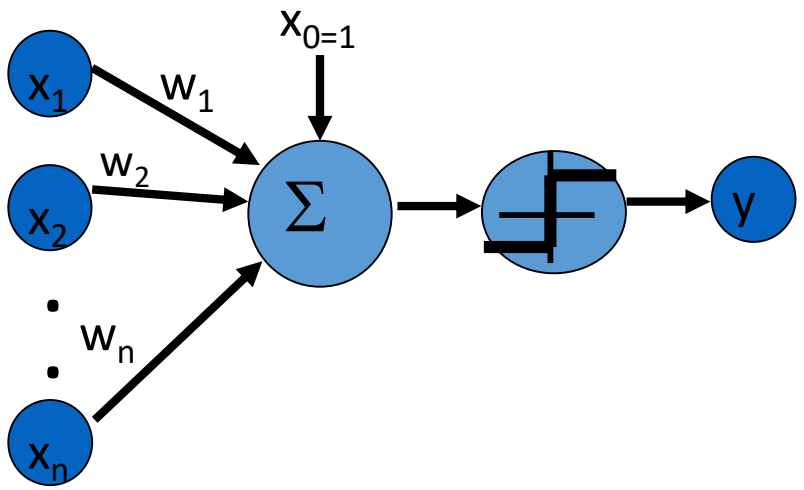
$$y_k = \begin{cases} 1 \; if \; w_1 x_1 + w_2 x_2 + b_k > C \\ 0 \; if \; w_1 x_1 + w_2 x_2 + b_k \leq C \end{cases}$$

Bias of ANN

- Bias in neural networks may be assumed as an extra perceptron (neuron) added to each layer, which can store the value of 1.
- This allows to “move” or translate the activation function so it doesn’t cross the origin, by adding a constant number.
- Without a bias, each perceptron (neuron) can only take the input and multiply it by a weight.
- For example, it would not be possible to input a value of 0 and output 2.
- In many cases, it is necessary to move the entire activation function to the left or right to generate the required output values – this is made possible by the bias.

Activation functions (Transfer function)

- **Transforms neuron's input into output.**
- **An activation function is a node that you add to the output layer or between two layers of any neural network. It is also known as the transfer function. It is used to determine the output of neural network layer in between 0 to 1 or -1 to 1 etc.**
- **A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.**



However, only *nonlinear* activation functions allow the networks to compute nontrivial problems using only a small number of nodes, and such activation functions are called nonlinearities.

Activation (Transfer) Functions

- ✓ Hard-Limit Function
- ✓ Linear function
- ✓ Sigmoid Function
- ✓ Rectified linear unit (RELU)
- ✓ Tanh Function
- ✓ Softmax Function

CHOOSING THE RIGHT ACTIVATION FUNCTION

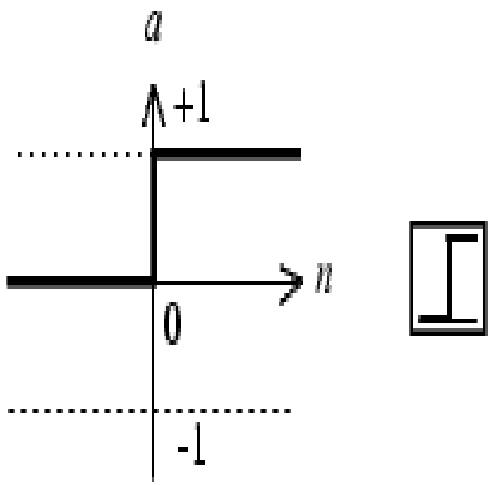
- **The basic rule of thumb is if you really don't know what activation function to use, then simply use *RELU* as it is a general activation function and is used in most cases these days.**
- **If your output is for binary classification then, *sigmoid function* is very natural choice for output layer.**
- **The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.**

Hard-Limit Functions

This function is used in Perceptrons*, to create neurons that make classification decisions.

*Perceptron networks have several limitations:

- the output values of a perceptron can take on only one of two values (0 or 1) because of the hard-limit transfer function.
- perceptrons can only classify linearly separable sets of vectors i.e. linear clusters.

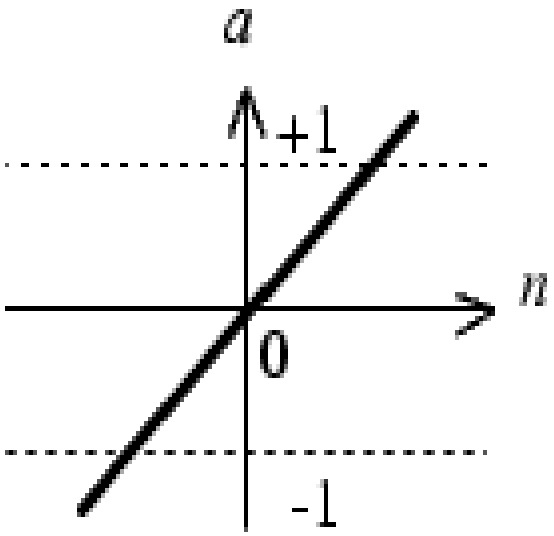


Hard-Limit Transfer Function

Linear Functions

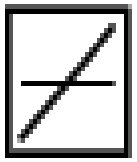
Neurons of this type are used as linear approximators in Linear* Filters.

*Limitation: Linear networks can only learn linear relationships between input and output vectors.



$a = purelin(n)$

Linear Transfer Function



Linear Function

- **Equation :** Linear function has the equation similar to as of a straight line i.e. **y = ax**
- No matter how many layers we have, if all are linear in nature, the final activation function of last layer is nothing but just a linear function of the input of first layer.
- **Range :** -inf to +inf
- **Uses : Linear activation function** is used at just one place i.e. output layer.
- **Issues :** If we will differentiate linear function to bring non-linearity, result will no more depend on *input "x"* and function will become constant, it won't introduce any ground-breaking behavior to our algorithm.
- **For example :** Calculation of price of a house is a regression problem. House price may have any big/small value, so we can apply linear activation at output layer. Even in this case neural net must have any non-linear function at hidden layers.

Sigmoid Function

- It is a function which is plotted as **'S'** shaped graph.
- **Equation :** $A = 1/(1 + e^{-x})$
- **Nature :** Non-linear. X values generally lies between -2 to 2, Y values are very steep. This means, small changes in x would also bring about large changes in the value of Y.
- **Value Range :** 0 to 1
- **Uses :** Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be ***1*** if value is greater than **0.5** and ***0*** otherwise.

Rectified linear unit (RELU)

- **RELU :-** Stands for *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of Neural network.
- **Equation :-** ***A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.
- **Value Range :-** [0, inf)
- **Nature :-** non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.
- **Uses :-** RELU is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.
- In simple words, RELU learns *much faster* than sigmoid and Tanh function.

Tanh Function

- The activation that works almost always better than sigmoid function is Tanh function also knows as **Tangent Hyperbolic function**. It's actually mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other.
- · **Equation :-** $f(x) = \tanh(x) = 2/(1 + e^{-2x}) - 1$

OR

- $\tanh(x) = 2 * \text{sigmoid}(2x) - 1$
- **Value Range :-** -1 to +1
- **Nature :-** non-linear
- **Uses :-** Usually used in hidden layers of a neural network as it's values lies between **-1 to 1** hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in *centering the data* by bringing mean close to 0. This makes learning for the next layer much easier.

Softmax Function

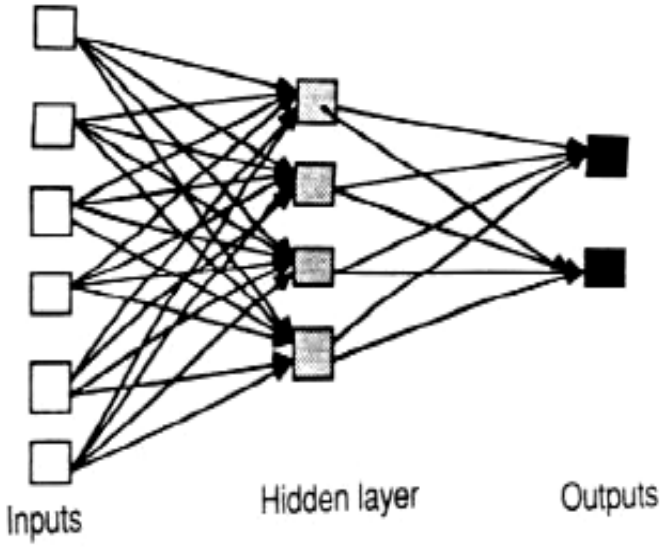
- The softmax function is also a type of sigmoid function but is handy when we are trying to handle classification problems.
- **Nature :-** non-linear
- **Uses :-** Usually used when trying to handle multiple classes. The softmax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs.
- **Output:-** The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

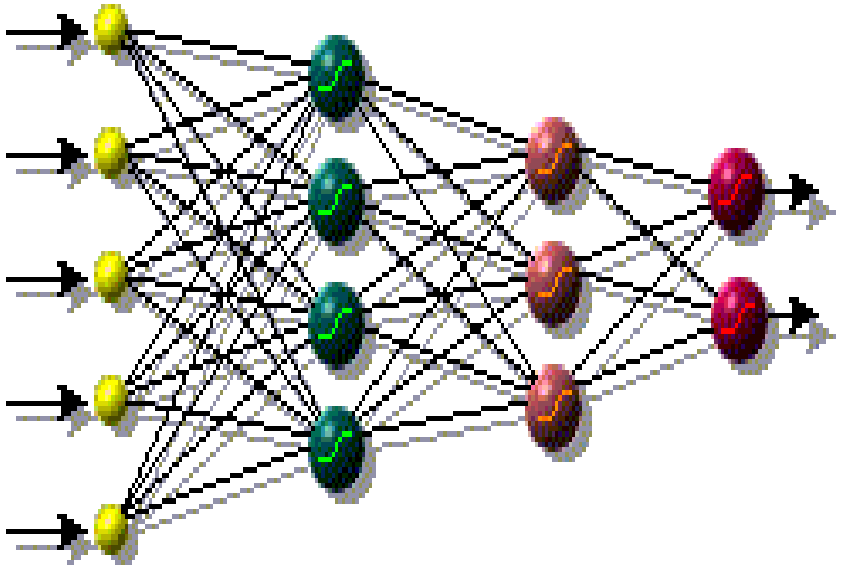
Types of Layers

- The input layer.
  - Introduces input values into the network.
  - No activation function or other processing.
- The hidden layer(s).
  - Perform classification of features
  - Two hidden layers are sufficient to solve any problem
  - Features imply more layers may be better
- The output layer.
  - Functionally just like the hidden layers
  - Outputs are passed on to the world outside the neural network.

The network

•Each ANN is composed of a collection of perceptrons grouped in layers. A typical structure is shown below.

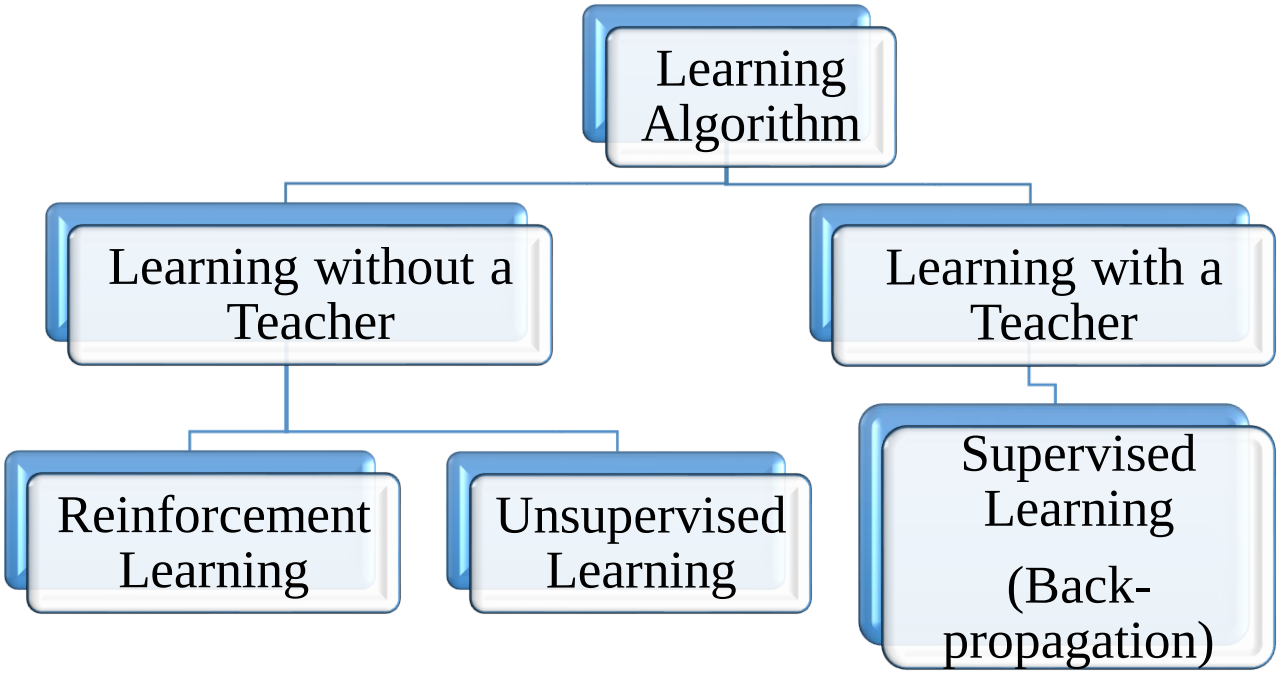




Note the three layers: input, intermediate (called the ***hidden layer***) and output. Several hidden layers can be placed between the input and output layers.

Learning Algorithm

The set of well defined rules for the solution of a learning problem is called a learning algorithm. Each learning algorithm differs from the other in the way in which the adjustment to a synaptic weight of a neuron is formulated.



Training (Learning)

- ***Unsupervised:*** If the target output is not available, the training is unsupervised. The network passes through a self-organizing process to produce outputs that are consistent with a group of similar inputs. Mostly used in data clustering.
- ***Supervised:*** When the input-output relation of the training scenario is available. The error in prediction is fed back to the network for updating its parameters so that the error is minimized.
- **Reinforcement:** It is learning by interacting with an environment. An RL (reinforcement learning) agent learns from the consequences of its actions, rather than from being explicitly taught and it selects its actions on basis of its past experiences (exploitation) and also by new choices (exploration), which is essentially *trial and error* learning.

Perceptron Training (Learning) Rule

- **One way to get an acceptable weight vector is to begin with random weights, then iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it misclassifies an example.** This process is repeated, iterating through the training examples as many as times needed until the perceptron *classifies* all training examples correctly.
- Weights are modified at each step according to the perceptron training rule, which revises the weight w_i associated with input x_i according to the rule:

$$w_i \leftarrow w_i + \Delta w_i$$

$$\text{where } \Delta w_i = \eta(t - o)\, x_i$$

- Here:

  t is target output value for the current training example

  o is perceptron output

  η is small constant (e.g., 0.1) called *learning rate*

Perceptron Training (Learning) Rule (cont.)

- The role of the **learning rate** is to moderate the degree to which weights are changed at each step. It is usually set to some small value (e.g. 0.1) and is sometimes made to decrease as the number of weight-tuning iterations increases.
- It is proved that the algorithm will converge
  - If training data is linearly separable
  - and η sufficiently small.
- If the data is not linearly separable, convergence is not assured.

TYPES OF NETWORK according to information flow

- ***Feed-forward network:*** Information is passed in one direction i.e. starting from input layer towards the output layer through the hidden layer. So it does not form any loop.
- ***Recurrent network:*** Information is passed in both forward and backward directions. There may be number of feedback loops in each layer.

TYPES OF NETWORK according to input output relation

- ***Regressive:*** gives a relation in terms of equation and the equation is generally linear(y=mx+c).
- ***Progressive or back-propagational:*** gives no equation rather than a trained network and its own internal mathematics. It is slower yet mostly used.

Combining Neural Network and Fuzzy Logic

Features	Fuzzy Systems	Neural Networks
Knowledge acquisition	Human Experts	Numerical data
Training method	Expertise knowledge needed	Learning by example
Reasoning	Heuristic, multi-valued	Algorithmic, parallel
Linguistic interface	Well-defined	None
Robustness	Very High	Very High

Combining Neural Network with Fuzzy Logic

- The performance of a fuzzy system can be increased by a large number of rules but there are some limitations.
- By combining the ANN and Fuzzy systems we can minimize the number of rules to get the same performance.
- Hence we can be able to train the connectionist fuzzy logic structure with the backpropagation learning rule.

BENEFITS OF ANN

- ***Usefulness:*** It is useful for pattern recognition, classification, generalization, abstraction and interpretation of incomplete and noisy inputs. (e.g. handwriting recognition, image recognition, voice and speech recognition, weather forecasting).
- ***Complicated characteristic Analysis:*** Providing some human characteristics to problem solving that are difficult to simulate using the logical, analytical techniques of expert systems and standard software technologies. (e.g. financial applications).
- ***Ability to solve new kinds of problems****:* ANNs are particularly effective at solving problems whose solutions are difficult, if not impossible, to define. This opened up a new range of decision support applications formerly either difficult or impossible to computerize.

BENEFITS OF ANN

- ***Robustness:*** ANNs tend to be more robust than their conventional counterparts. They have the ability to cope with imcomplete or fuzzy data. ANNs can be very tolerant of faults if properly implemented.
- ***Fast processing speed:*** Because they consist of a large number of massively interconnected processing units, all **operating in parallel** on the same problem, ANNs can potentially operate at considerable speed (when implemented on parallel processors).
- ***Flexibility and ease of maintenance:*** ANNs are very flexible in adapting their behavior to new and changing environments. They are also easier to maintain, with some having the ability to learn from experience to improve their own performance.

Limitations of ANNs

- ANNs do not produce an *explicit model* even though new cases can be fed into it and new results obtained.
- *ANNs lack explanation capabilities*. Justifications for results is difficult to obtain because the connection weights usually do not have obvious interpretations.

SOME ANN APPLICATIONS

- Tax form processing to identify tax fraud
- Enhancing auditing by finding irregularites
- Bankruptcy prediction
- Customer credit scoring
- Loan approvals
- Credit card approval and fraud detection
- Financial prediction
- Energy forecasting
- Computer access security (intrusion detection and classification of attacks)
- Fraud detection in mobile telecommunication networks
- Bio-medical applications

Thank You

MULTILAYER NETWORKS AND BACKPROPOGATION ALGORITHM

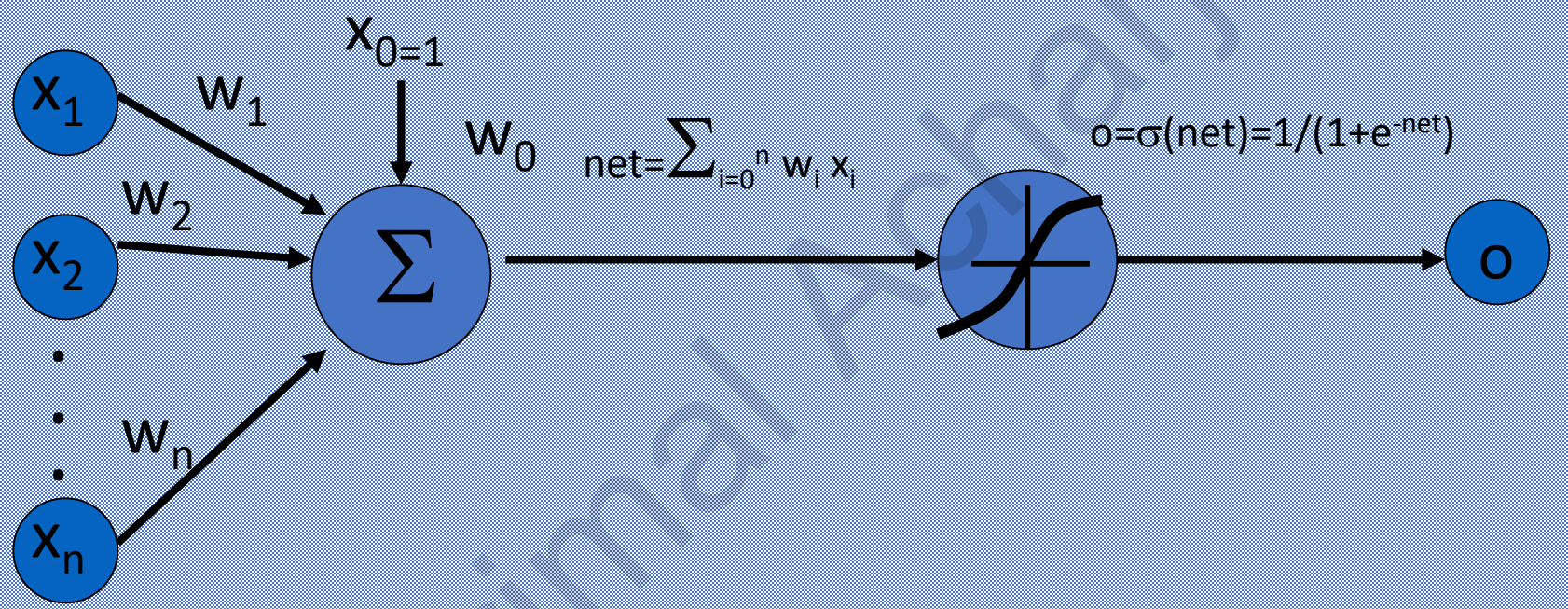
- Single perceptron can only express linear decision surfaces.
- In contrast, the kind of multilayer networks trained by the backpropagation algorithm are capable of expressing a rich variety of **nonlinear** decision surfaces.

➢ **A Differentiable Threshold Unit**

- Perceptron : not differentiable -> can't use gradient descent
- Linear Unit : multi-layers of linear units -> still produce only linear function
- Sigmoid Unit : *smoothed, differentiable threshold function*

Prof. Parimal Acharjee

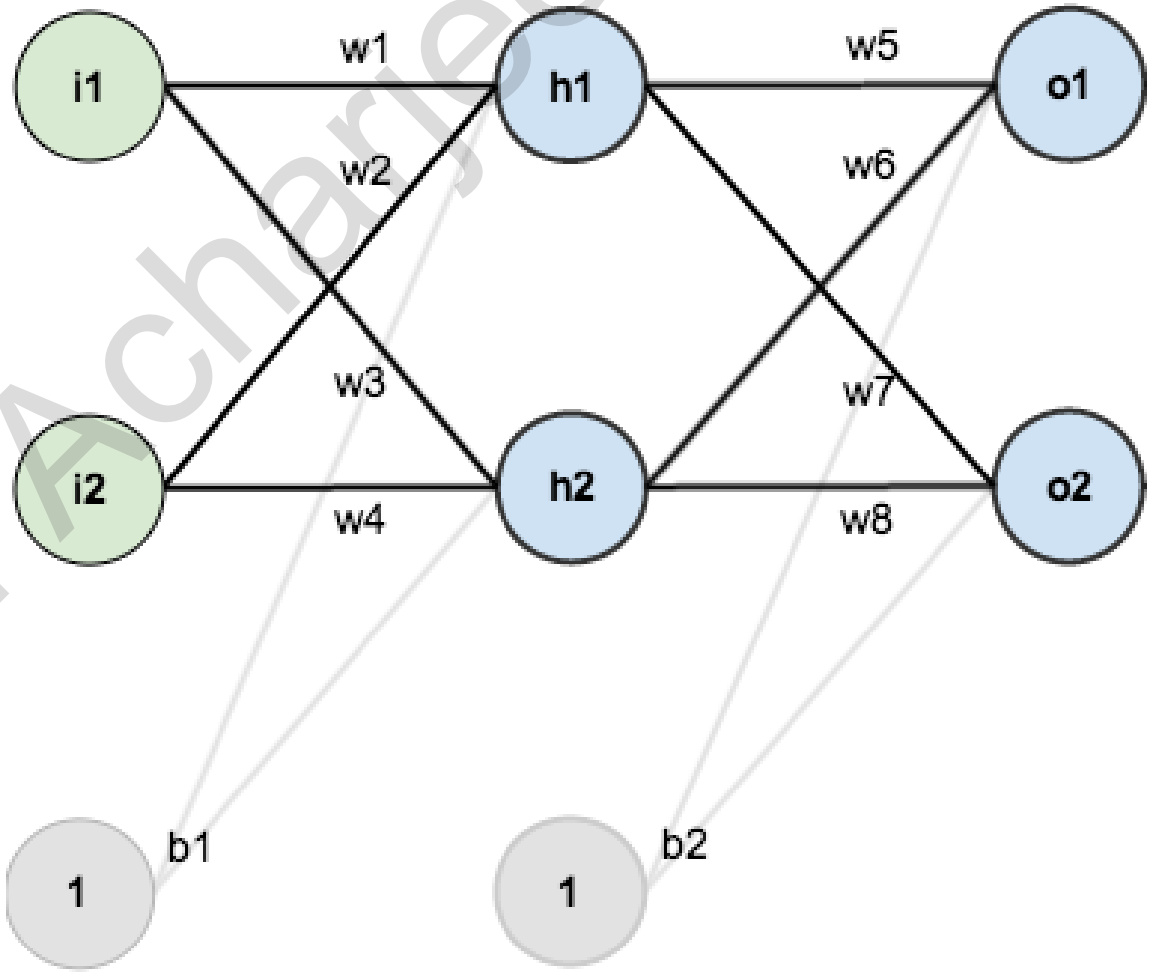
Sigmoid Unit

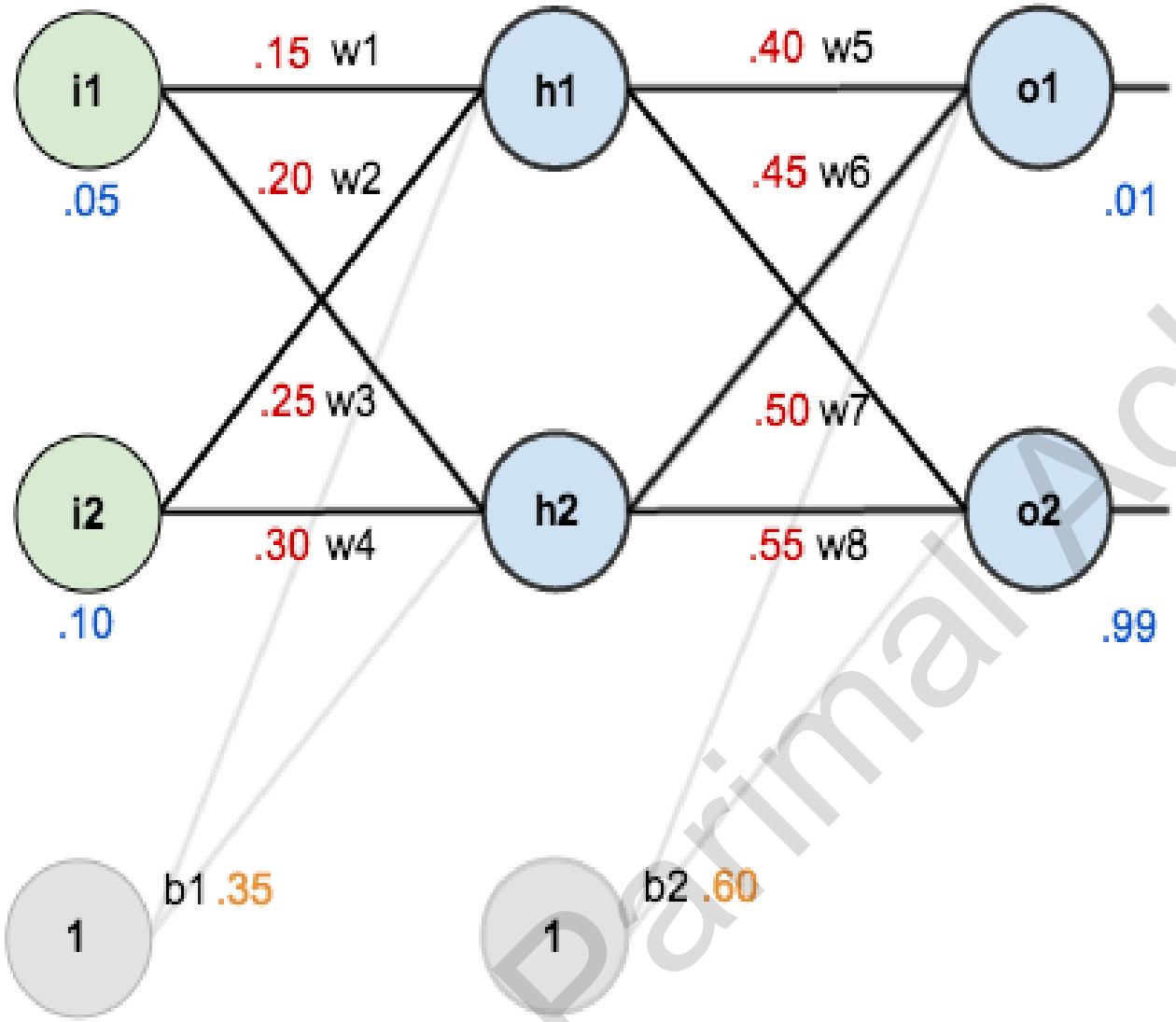


$\sigma(x)$ is the sigmoid function: $1/(1+e^{-x})$

Example of Backpropogation

- A neural network with two inputs,
- two hidden neurons, two output neurons.
- Additionally, the hidden and output neurons will include a bias.





Initial weights, the biases, and training inputs/outputs

- In this example, consider a single training set
- Given inputs 0.05 and 0.10.
- Desired output 0.01 and 0.99.
- Goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.

The Forward Pass

- At the beginning, predicts the weights and biases.
- Get inputs. Here, Two Inputs: 0.05 and 0.10.
- Those inputs forward through the network.
- Figure out the *total net input* to each hidden layer neuron.
- *Apply* an *activation function.*
- *In this example,* the *logistic function is used.*
- Repeat the process with the output layer neurons.

Total net input for hidden layers and outputs of hidden layers

Net Input for hidden layers 1 (h_1)

$$= w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1$$

$$= 0.3775$$

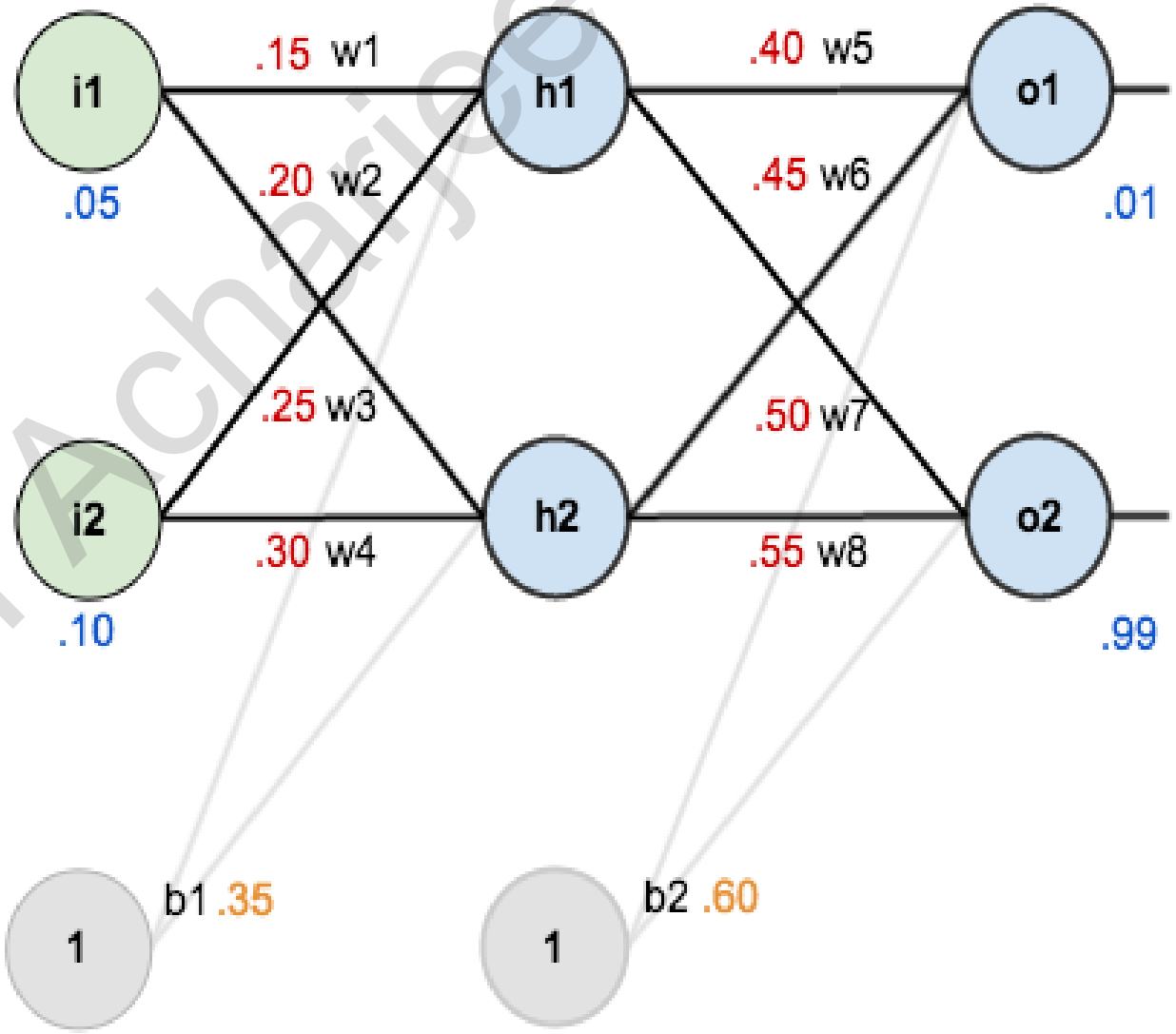
Apply the logistic function (sigmoid function) to get the output of h_1

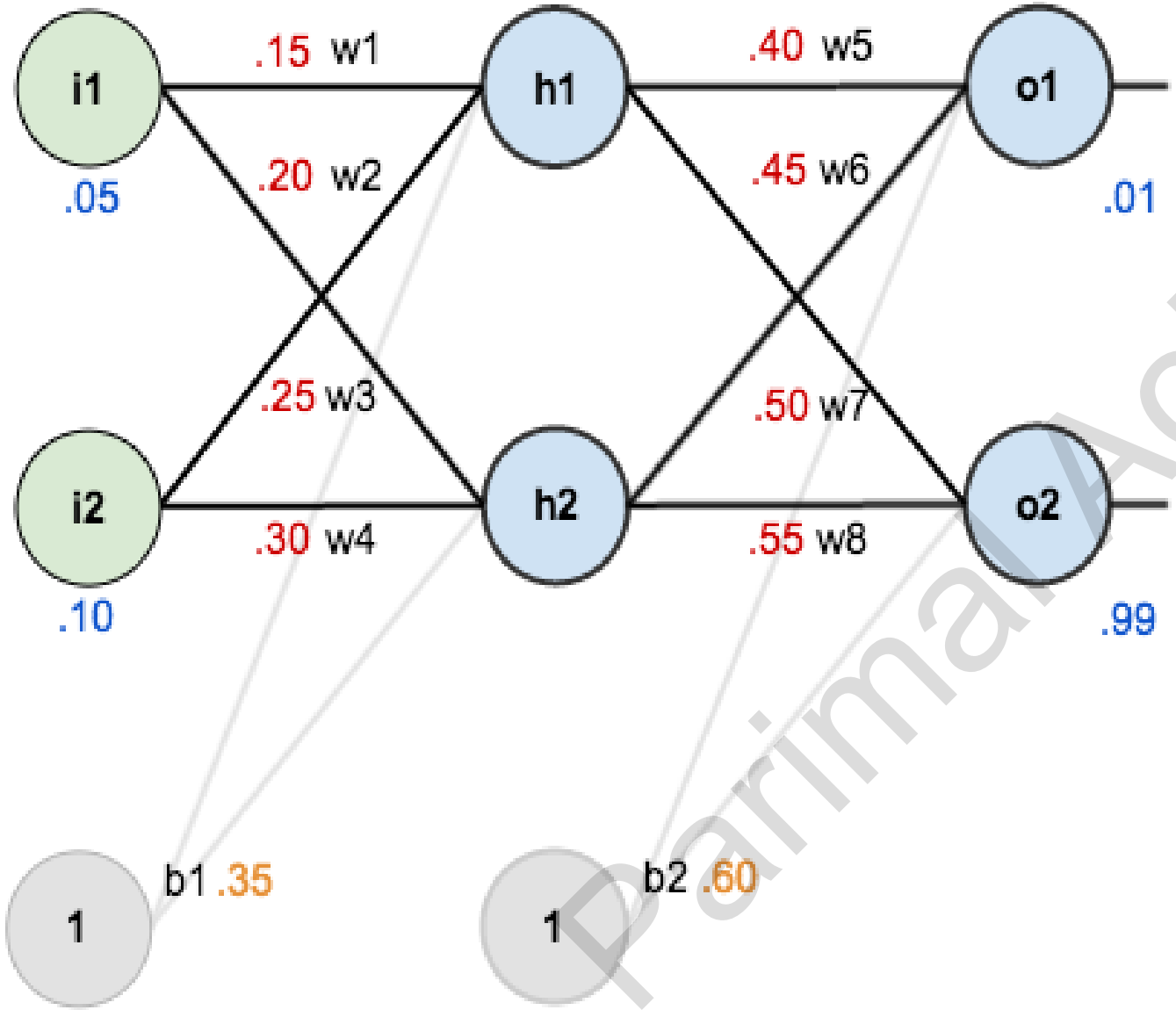
$$out_{h1} = \frac{1}{1 + e^{-net_{h1}}} = \frac{1}{1 + e^{-0.3775}}$$

$$= 0.593269992$$

Follow the same process for h_2

out_{h2}=0.596884378





Output from the hidden layer neurons will be treated as input.
Apply same process for the output layer neurons.

We have,

$$out_{h1} = 0.593269992$$
$$out_{h2} = 0.5968884378$$

$$net_{01} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{01} = 0.4 * 0.593269992 + 0.45 * 0.5968884378 + 0.6*1$$

$$= 1.105905967$$

$$out_{01} = \frac{1}{1+e^{-net_{01}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

Carrying out the same process for O_2

$$out_{02} = 0.772928465$$

Determine Error

- Apply squared error function to determine the error for each output neuron.
- Add output errors to obtain the total error (E_{total})

$$E_{total} = \sum_{i=1}^{n} \frac{1}{2}(target_i - out_i)^2$$

- In some papers, target is denoted as ideal.
- Output is mentioned as actual.
- Half is included so that exponent is cancelled when we differentiate later on.
- The result is eventually multiplied by a learning rate anyway so it doesn't matter that we introduce a constant here.

Total Error

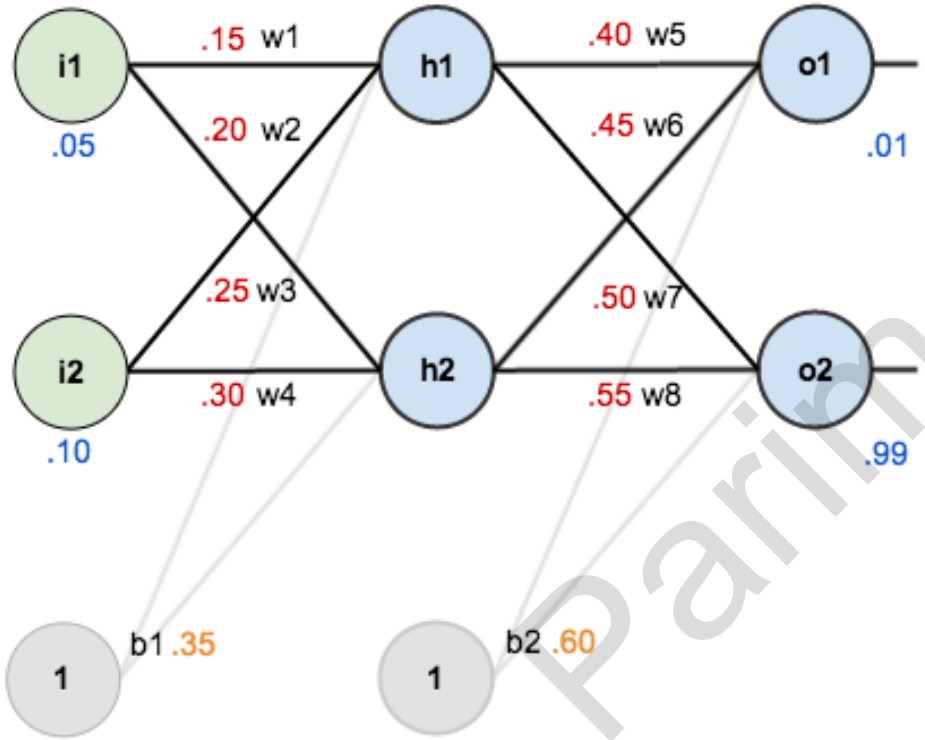
- Target output for O_1: $target_{01}$=0.01
- Neural network output for O_1: out_{01}=0.75136507
- Error for O_1 (E_{01})
- $E_{01} = \frac{1}{2}(target_{01} - out_{01})^2 = \frac{1}{2}(0.01 - 0.75136507)^2$=0.274811083
- Target output for O_2: $target_{02}$=0.99
- Neural network output for O_1: out_{02}=0.772928465
- Error for O_2 (E_{02})
- E_{02}=0.023560026
- Total Error (E_{total})= E_{01}+E_{02}=0.298371109

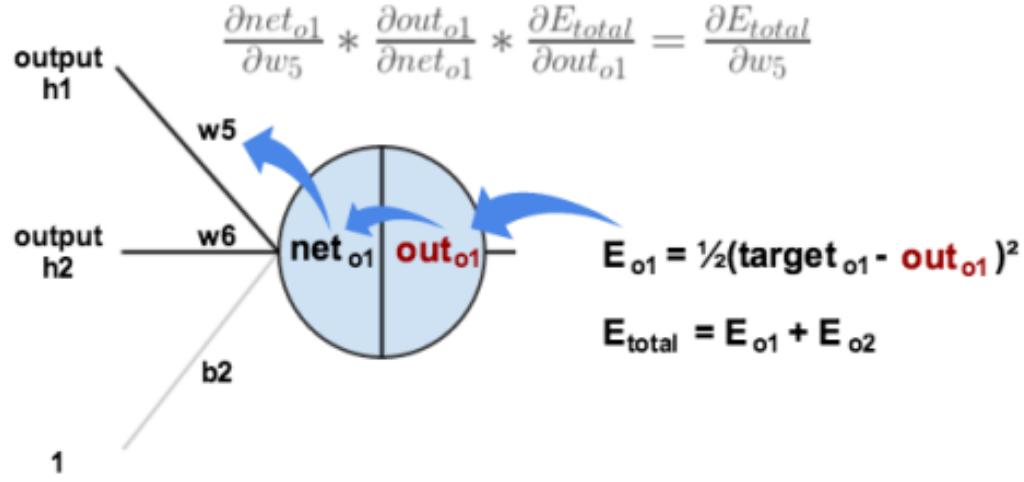
The Backwards Pass

- Minimize the error for each output neuron and the network as a whole.
- Obtain the actual output to be closer the target output.
- To achieve the goal, update each of the weights in the network.
- Errors will be used in the backward layers to update weights.
- Consider w_5 of the output layer to be updated using the total error (E_{total}) of the network
- Need differential of E_{total} with respect to w_5
- $\frac{\partial E_{total}}{\partial w_5}$: the partial derivative of E_{total} with respect to w_5

OR the gradient with respect to w_5

Visualization of Backward Pass





Determine $\frac{\partial E_{total}}{\partial out_{01}}$

Calculate the total error change with respect to the output.

$$E_{total} = \frac{1}{2}(target_{01} - out_{01})^2 + \frac{1}{2}(target_{02} - out_{02})^2$$

$$\frac{\partial E_{total}}{\partial out_{01}} = 2 * \frac{1}{2}(target_{01} - out_{01})^{2-1} * (-1) + 0$$

$$\frac{\partial E_{total}}{\partial out_{01}} = -(target_{01} - out_{01}) = -(0.01 - 0.75136507) = 0.74136507$$

Determination of $\frac{\partial E_{total}}{\partial w_5}$

Sigmoid function is used. We have, $out_{01} = \frac{1}{1+e^{-net_{01}}}$

$$\frac{\partial out_{01}}{\partial net_{01}} = out_{01} * (1 - out_{01}) = 0.75136507 * (1 - 0.75136507)$$

$$= 0.186815602$$

$$net_{01} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{01}}{\partial w_5} = out_{h1} + 0 + 0 = out_{h1} = 0.593269992$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{01}} * \frac{\partial out_{01}}{\partial net_{01}} * \frac{\partial net_{01}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{01} - out_{01}) * out_{01} * (1 - out_{01}) * out_{h1}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

Update weights

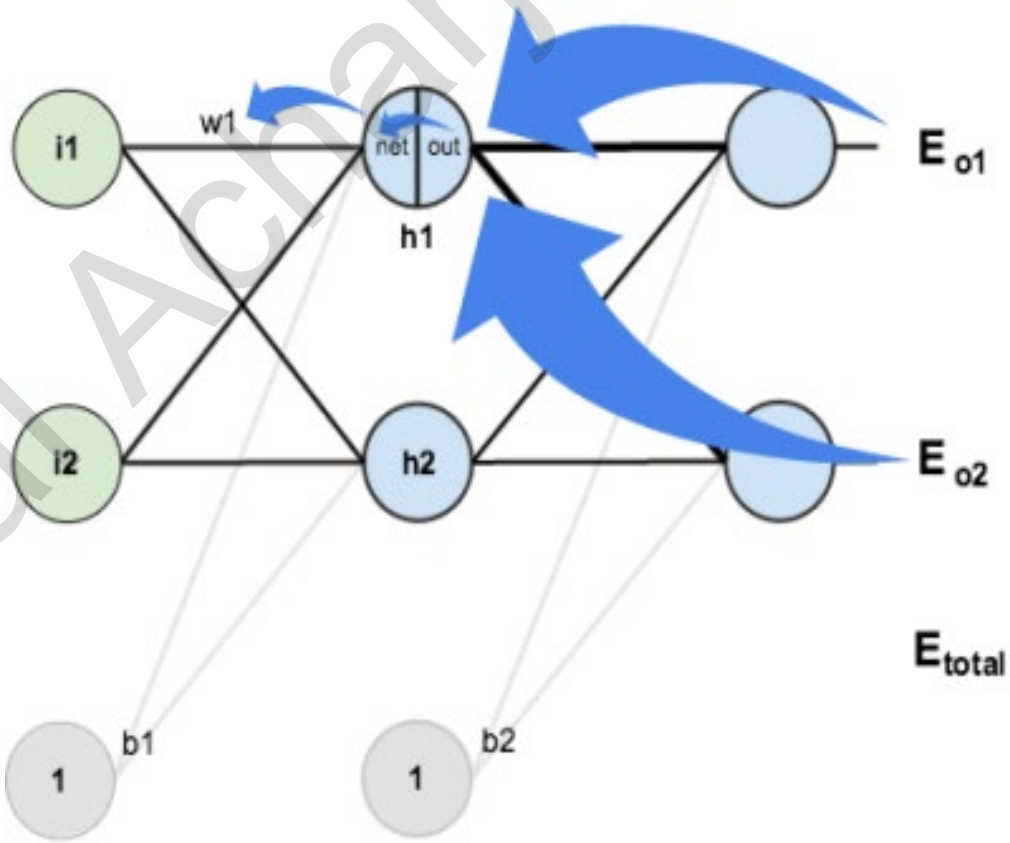
- Assume learning rate (η)=0.5
- Update w_5 as follows:
- $w_5^{updated} = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5}$=0.4-0.5*0.082167041=0.35891648
- Follow the same procedure to get the new weights of w_6, w_7, w_8
- $w_6^{updated} = 0.408666186$
- $w_7^{updated} = 0.511301270$
- $w_8^{updated} = 0.561370121$

Backward Pass for hidden layer

- Update w_1, w_2, w_3, w_4 through backward pass.
- Follow the same procedure as like as output layer
- The output of each hidden layer neuron contributes to all outputs.
- out_{h1} affects both out_1 and out_2
- In other words, out_{h1} affects both E_{01} and E_{02}.
- $\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{01}}{\partial out_{h1}} + \frac{\partial E_{02}}{\partial out_{h1}}$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



$E_{total} = E_{o1} + E_{o2}$

Updated weights and results

- Updated (new) weight values: $w_1 = 0.14978071$; $w_2 = 0.19956143$ $w_3 = 0.24975114$; $w_4 = 0.29950229$
- After updating weights, calculate total error again.
- Now, calculated total error= 0.291027924
- Earlier iteration (or step) total error was 0.298371109
- The improvement is trivial.
- Continue the process 10,000 times.
- Total error reduced to 0.0000351085
- Result : out_{01} =0.015912196 (where $target_{01}$=0.01) and out_{02}= 0.984065734 (where $target_{02}$= 0.99)

Thank you!

Fuzzy Logic

Parimal Acharjee
Professor
Electrical Engineering Department
NIT Durgapur

You should park the car in reverse gear at a speed of 6.7 Km./hr. keeping the accelerator on for 9.3 Sec.

Crisp thinking

How do I park the car?

At a little speed backwards for a few second

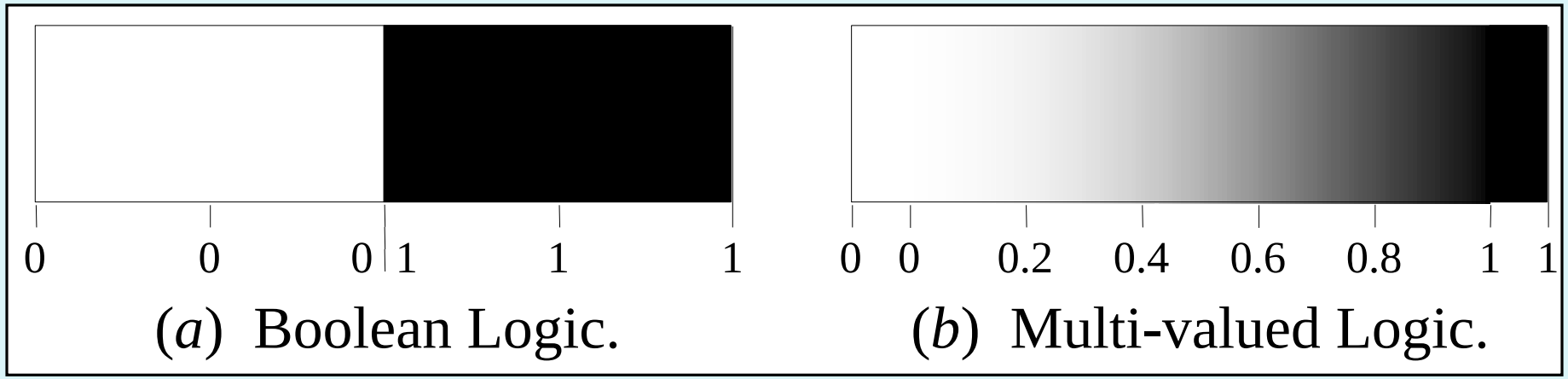
Fuzzy thinking

What is Fuzzy Thinking

- Experts rely on common sense when they solve problems.
- Fuzzy logic is not logic that is fuzzy, but logic that is used to describe fuzziness. Fuzzy logic is the theory of fuzzy sets, sets that calibrate vagueness.
- Fuzzy logic is based on the idea that all things admit of degrees. Temperature, height, speed, distance, beauty – all come on a sliding scale. The motor is running *really hot*. Tom is a *very tall* guy.

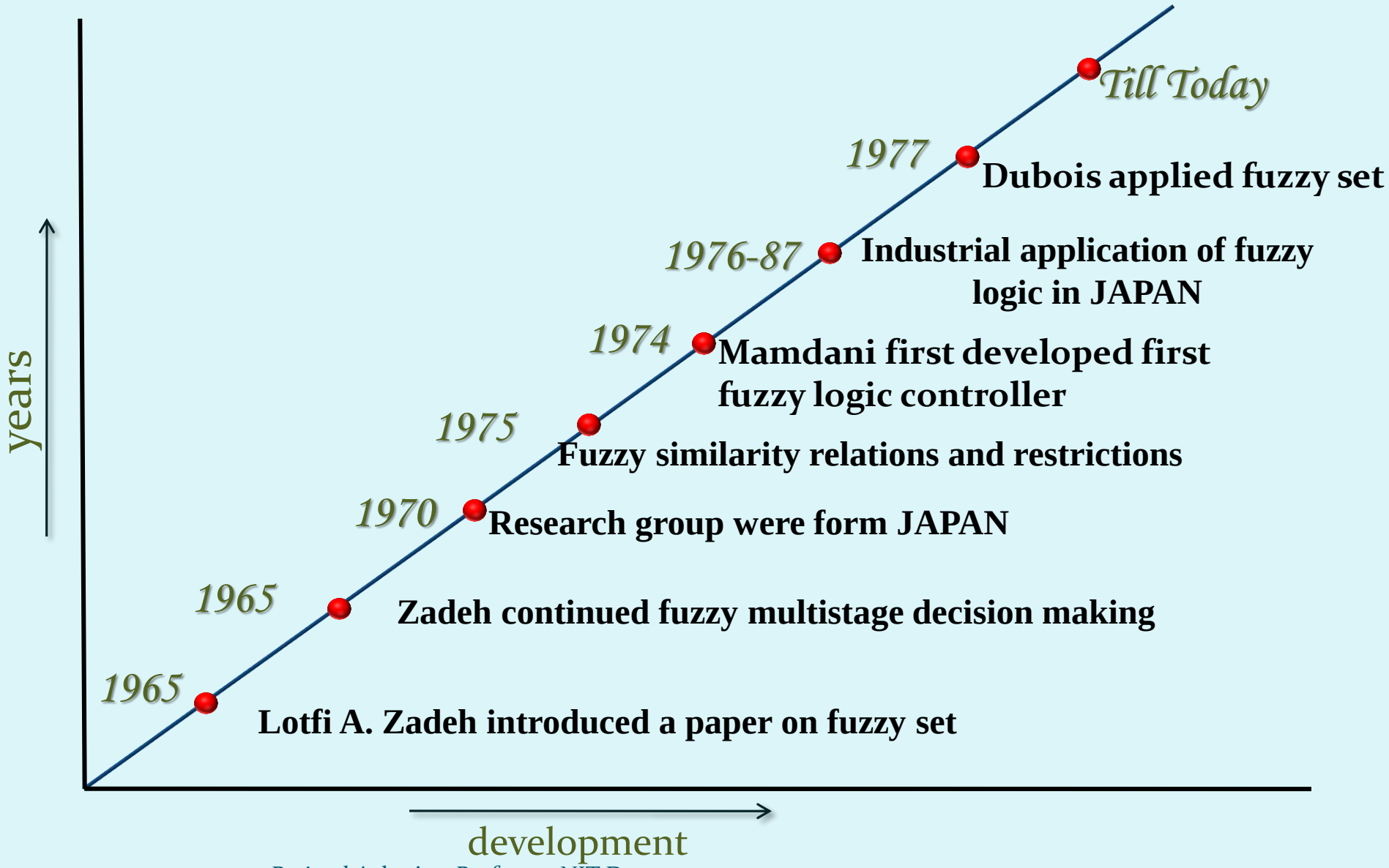
- Boolean logic uses sharp distinctions. It forces us to draw lines between members of a class and non-members. For instance, we may say, Tom is tall because his height is 181 cm. If we drew a line at 180 cm, we would find that David, who is 179 cm, is small. Is David really a small man or we have just drawn an arbitrary line in the sand?
- Fuzzy logic reflects how people think. It attempts to model our sense of words, our decision making and our common sense. As a result, it is leading to new, more human, intelligent systems.

Range of logical values in Boolean and Fuzzy logic



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

Historical Background



Advantages of Fuzzy Logic

- Requires neither electronic expertise nor large development time.
- Using ‘If…Then’ rules to achieve higher degree automation.
- Due to large load disturbance and system nonlinearity where conventional controller fails, there fuzzy controller is used.
- Any continuous nonlinear process can be controlled by Fuzzy Controller provided we have the sufficient knowledge.

Fuzzy Sets

- The concept of a **set** is fundamental to mathematics.

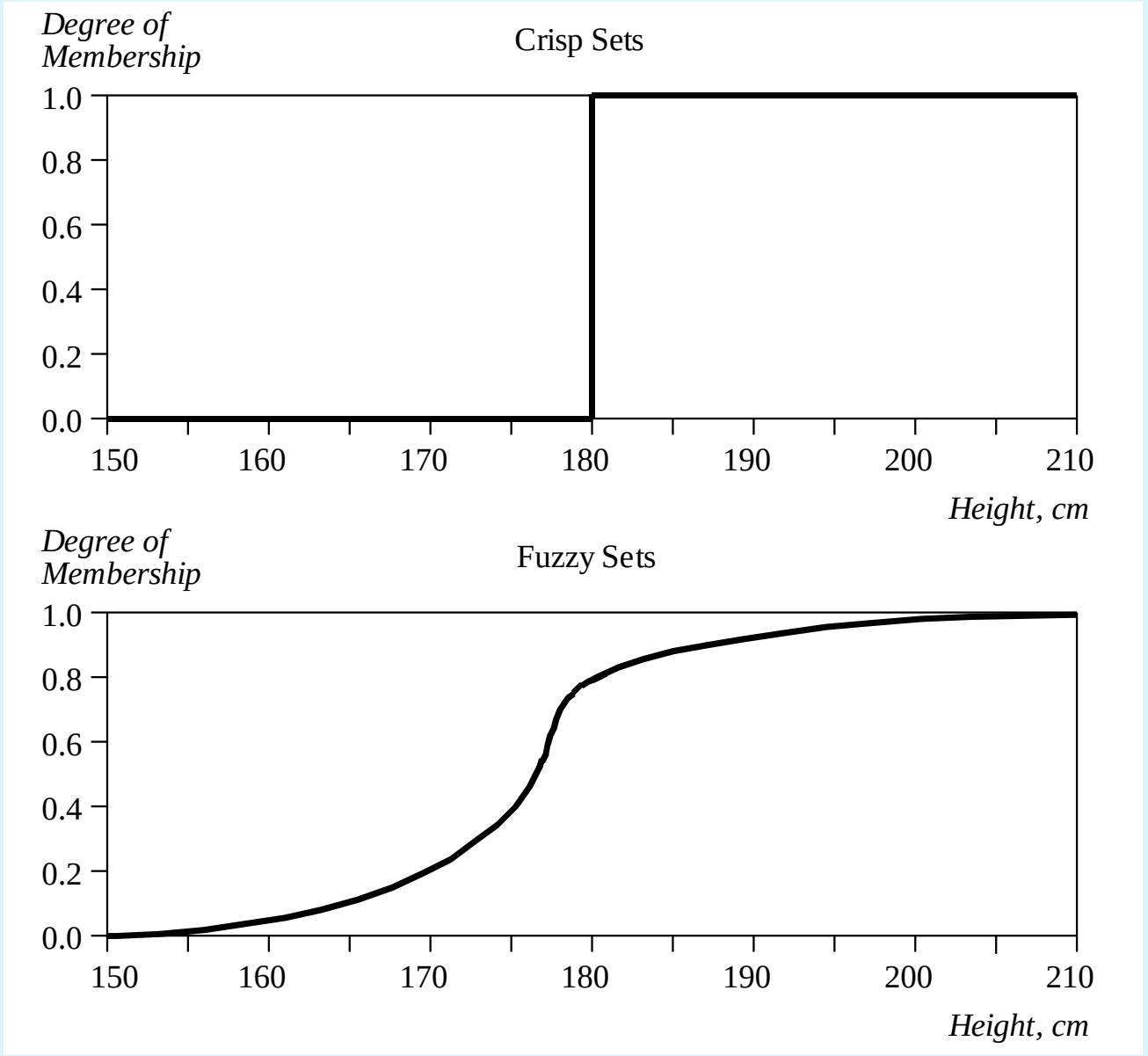
- However, our own language is also the supreme expression of sets. For example, *car* indicates the *set of cars*. When we say *a car*, we mean one out of the set of cars.

- Crisp set theory is governed by a logic that uses one of only two values: true or false.
- This logic cannot represent vague concepts, and therefore fails to give the answers on the paradoxes.
- The basic idea of the fuzzy set theory is that an element belongs to a fuzzy set with a certain degree of membership. Thus, a proposition is not either true or false, but may be partly true (or partly false) to any degree.
- This degree is usually taken as a real number in the interval [0,1].

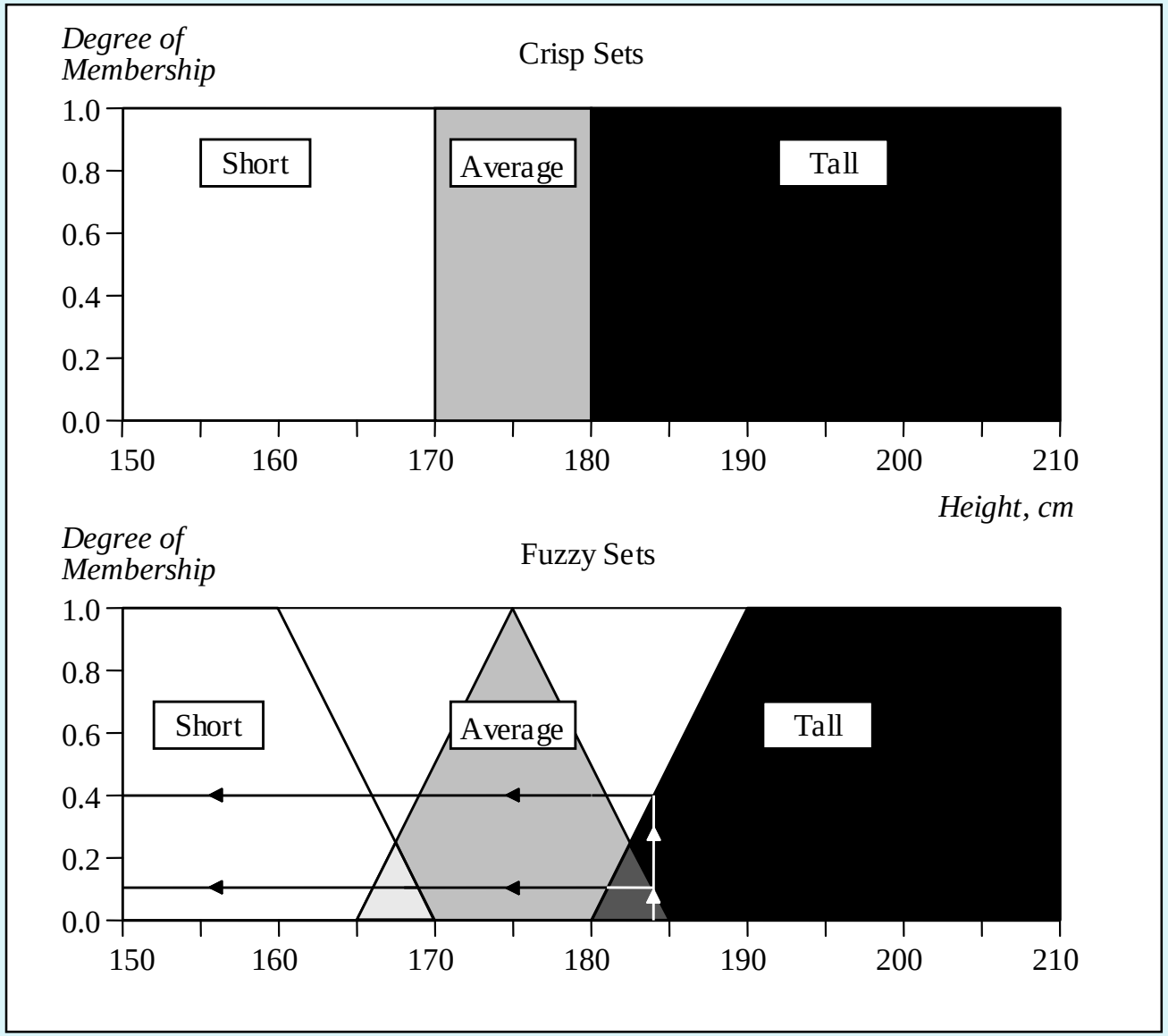
- The classical example in fuzzy sets is *tall men*. The elements of the fuzzy set "tall men" are all men, but their degrees of membership depend on their height.

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 and Fuzzy sets of ‘tall men’

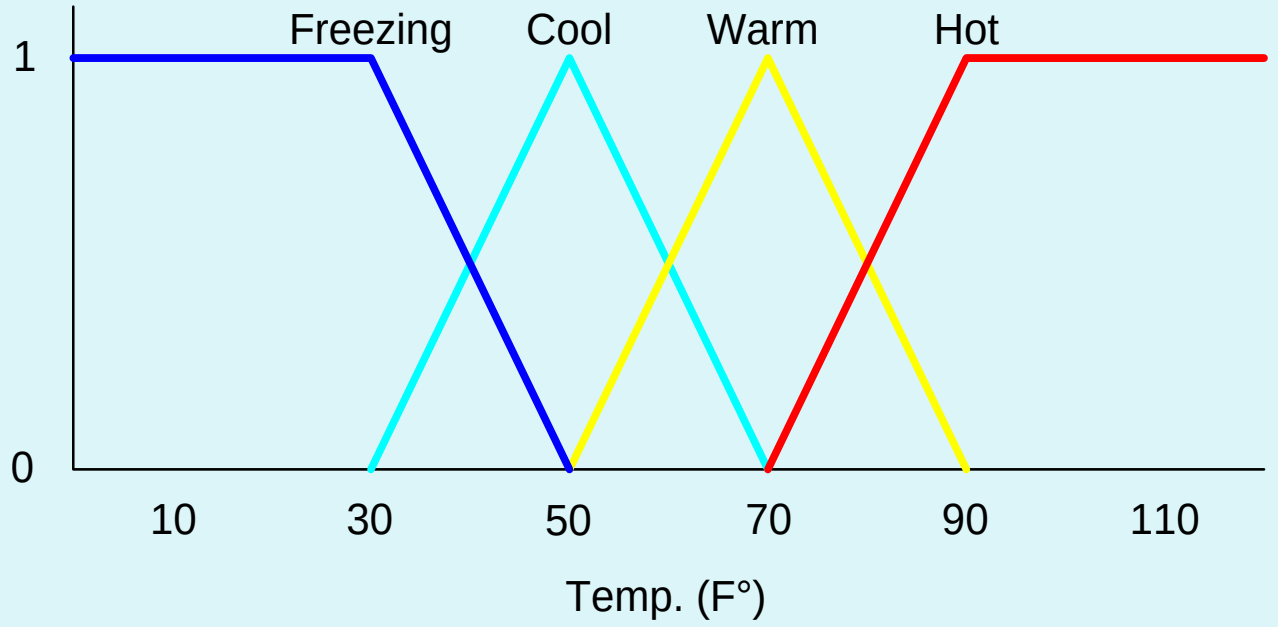


Crisp and Fuzzy sets of shorts, average and tall men



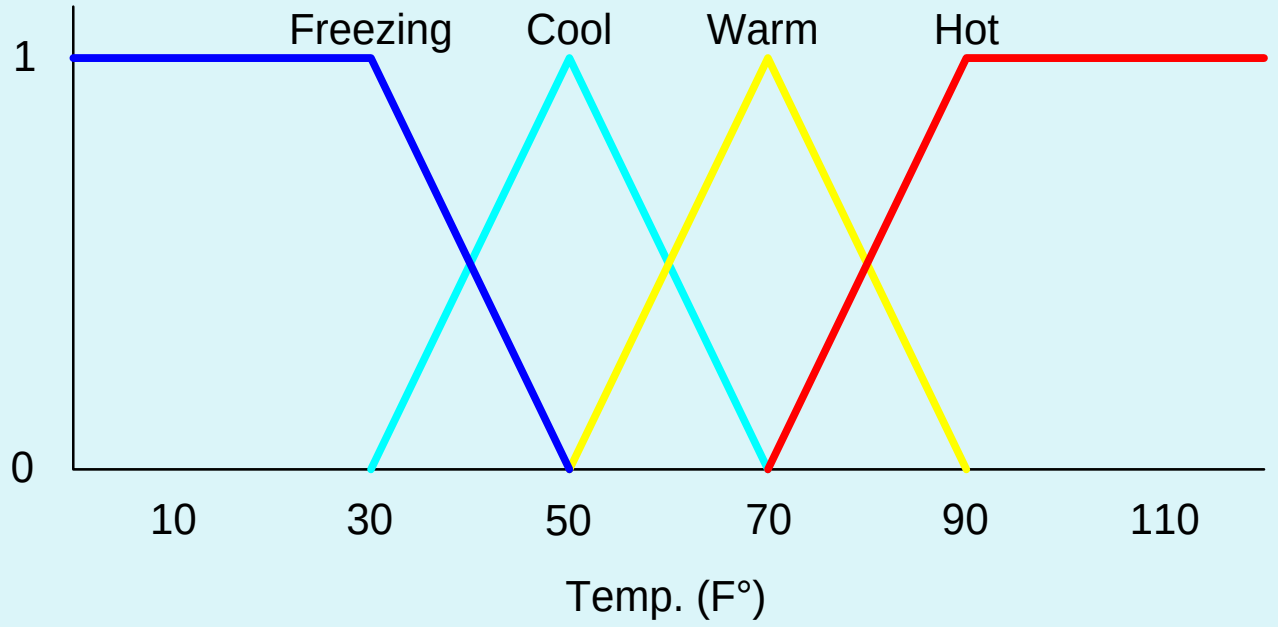
Membership function

- Temp: {Freezing, Cool, Warm, Hot}
- Degree of Truth or "Membership"



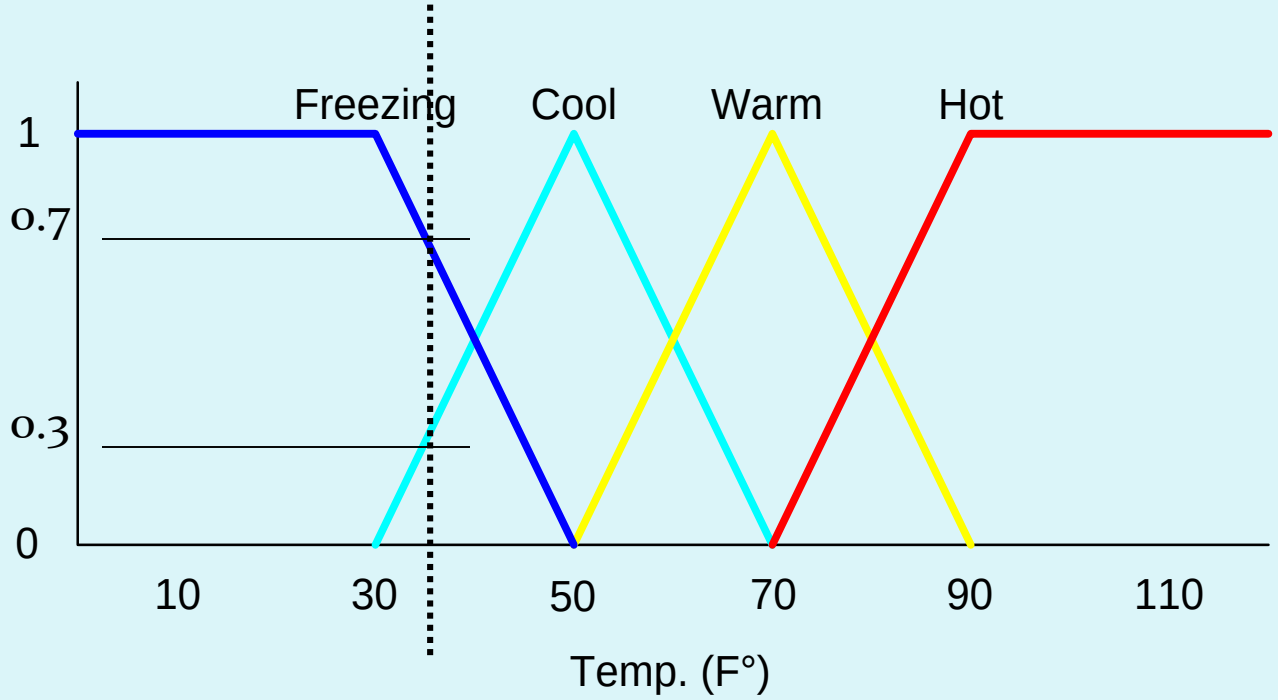
Membership function

- Temp: {Freezing, Cool, Warm, Hot}
- Degree of Truth or "Membership"

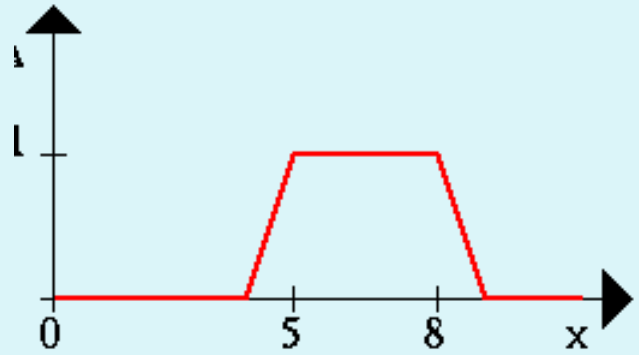


Membership Functions

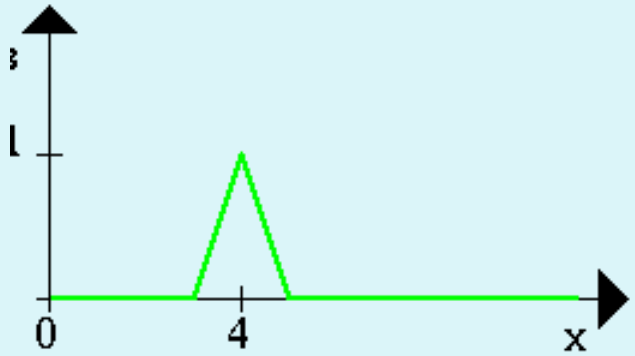
- How cool is 36 F° ?
- It is 30% Cool and 70% Freezing



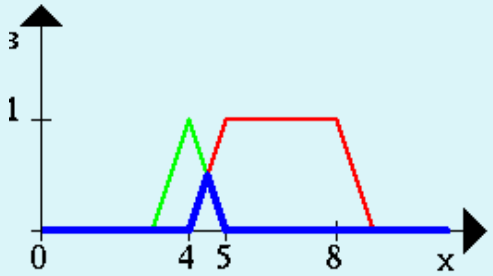
Operations



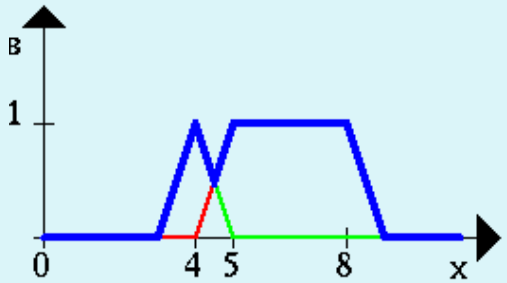
A



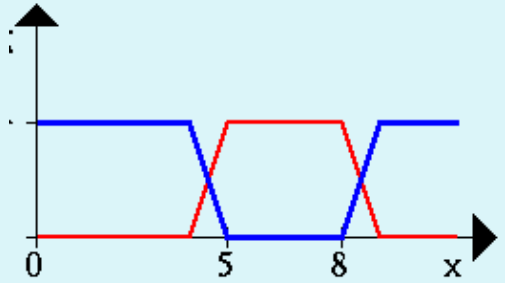
B



$A \wedge B$



$A \vee B$



$\neg A$

Controller Structure

- Fuzzification
  - Scales and maps input variables to fuzzy sets
- Inference Mechanism
  - Approximate reasoning
  - Deduces the control action
- Defuzzification
  - Convert fuzzy output values to control signals

Fuzzy Rules

In 1973, **Lotfi Zadeh** published his second most influential paper. This paper outlined a new approach to analysis of complex systems, in which Zadeh suggested capturing human knowledge in fuzzy rules.

What is Fuzzy Rules

A fuzzy rule can be defined as a conditional statement in the form:

IF x is A
THEN y is B

where x and y are linguistic variables; and A and B are linguistic values determined by fuzzy sets on the universe of discourses X and Y, respectively.

A fuzzy rule can have multiple antecedents, for example:

IF project_duration is long
AND project_staffing is large
AND project_funding is inadequate
THEN risk is high

IF service is excellent
OR food is delicious
THEN tip is generous

The consequent of a fuzzy rule can also include multiple parts, for instance:

IF temperature is hot
THEN hot_water is reduced;
cold_water is increased

Rule Base

Air Temperature

- Set cold {50, 0, 0}
- Set cool {65, 55, 45}
- Set just right {70, 65, 60}
- Set warm {85, 75, 65}
- Set hot {∞, 90, 80}

Fan Speed

- Set stop {0, 0, 0}
- Set slow {50, 30, 10}
- Set medium {60, 50, 40}
- Set fast {90, 70, 50}
- Set blast {∞, 100, 80}

Rules

default:

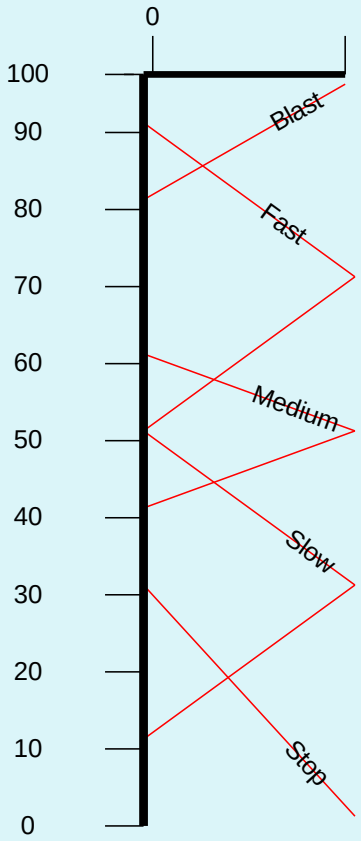
The truth of any statement is a matter of degree

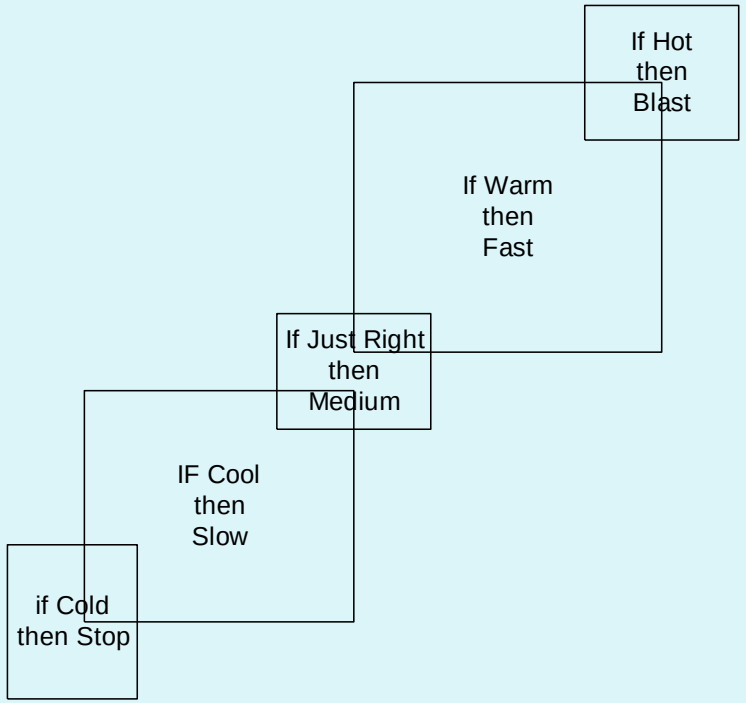
Membership function is a curve of the degree of truth of a given input value

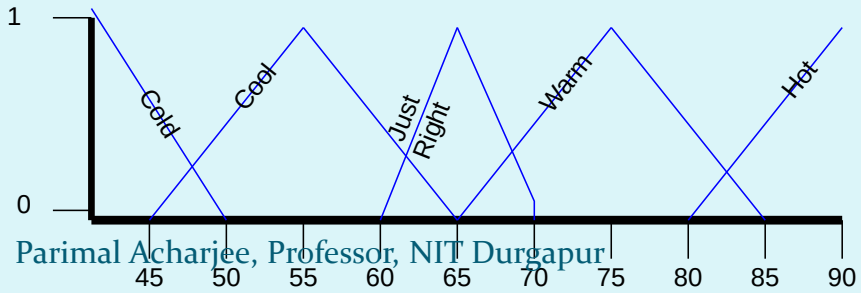
Air Conditioning Controller Example:

- IF Cold then Stop
- If Cool then Slow
- If OK then Medium
- If Warm then Fast
- IF Hot then Blast

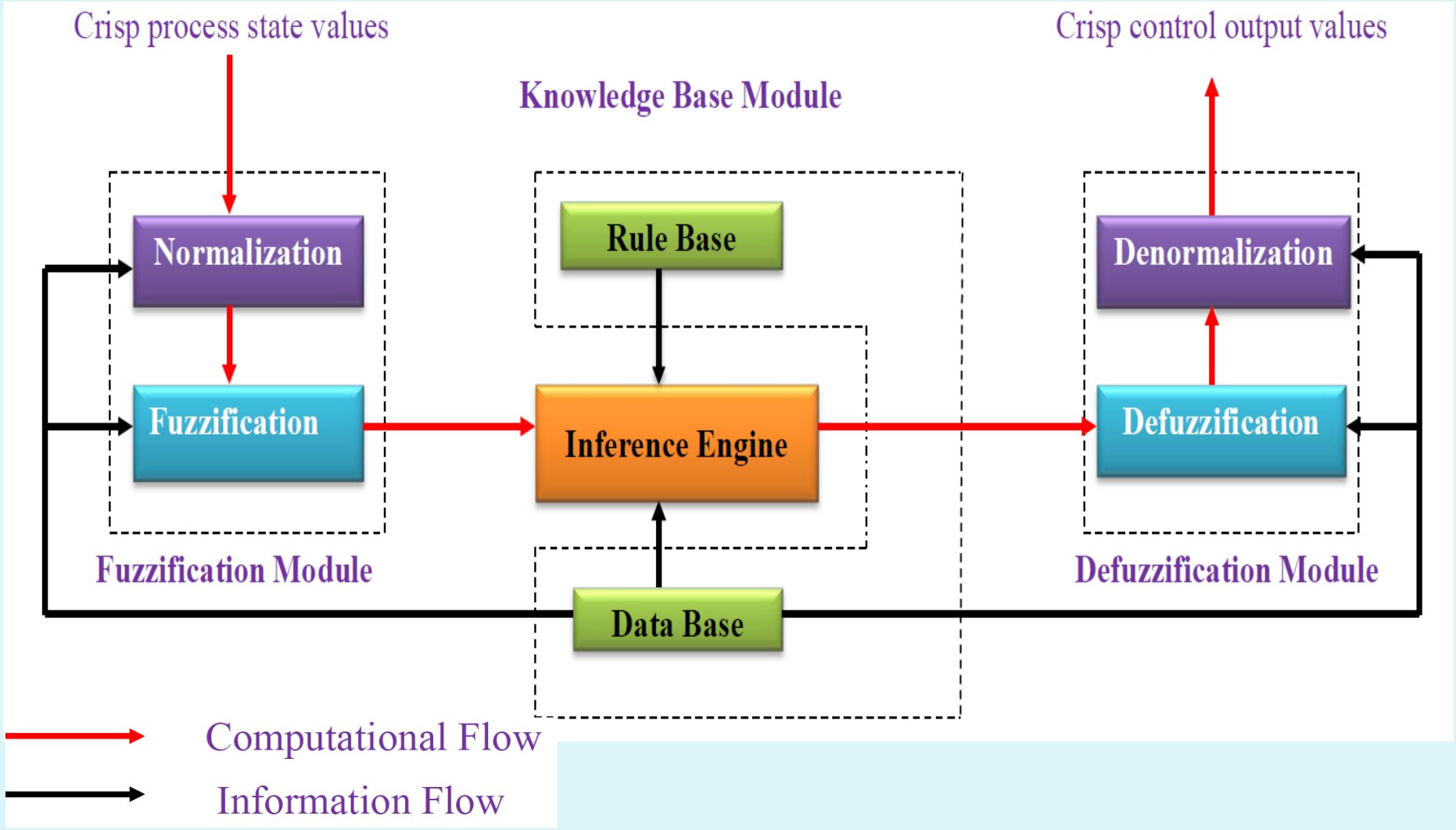
Fuzzy Air Conditioner







Structure of FKBC



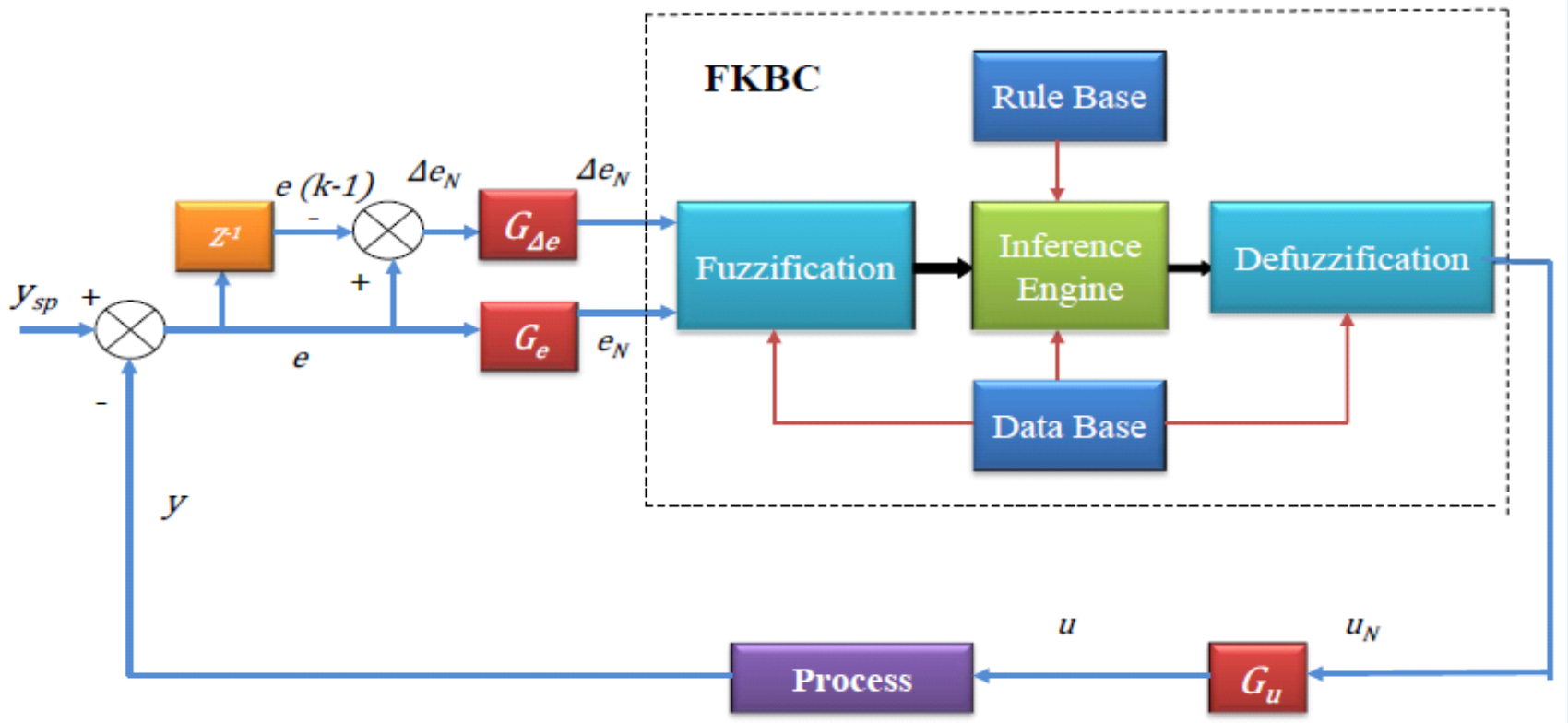
Structure of Fuzzy knowledge Based Controller

Parimal Acharjee, Professor, NIT Durgapur

Structure of Fuzzy PD controller

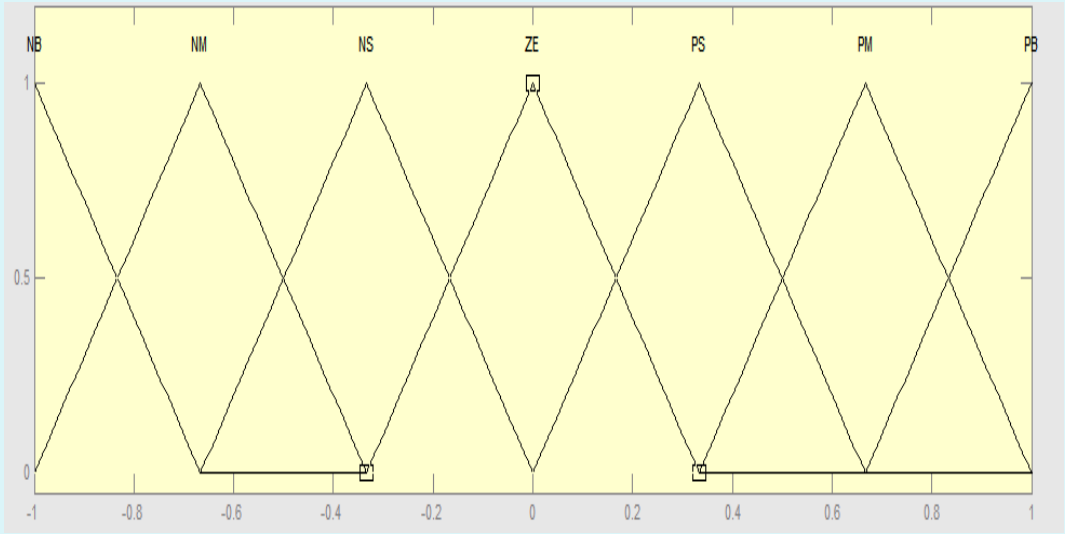
- In discrete domain PD Controller takes the following expression (considering the position form).

$$u(k) = K_P e(k) + K_D \Delta e(k)$$



In our model we have taken the Scaling Factors as Ge=1, Gde=5, Gu=5.

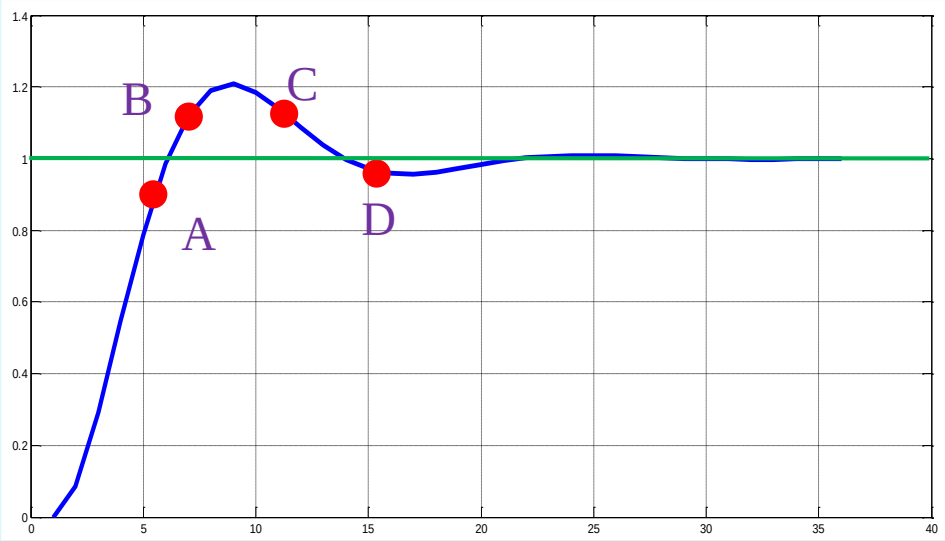
Membership Functions (Mamdani model)



MF for error and change of error

- For the Fuzzy PD controller we have considered 7 input and 7 output fuzzy sets as ‘Negative Big’ (NB), ‘Negative Medium’ (NM), ‘Negative Small’ (NS), ‘Zero’ (ZE), ‘Positive Small’ (PS), ‘Positive Medium’ (PM), and ‘Positive Big’ (PB).

Fuzzy Linguistic Notation and Rule Base



if Δe is NB and e is NB then u is NB.

if Δe is PS and e is NS then u is ZE.

if Δe is PB and e is PS then u is PB.

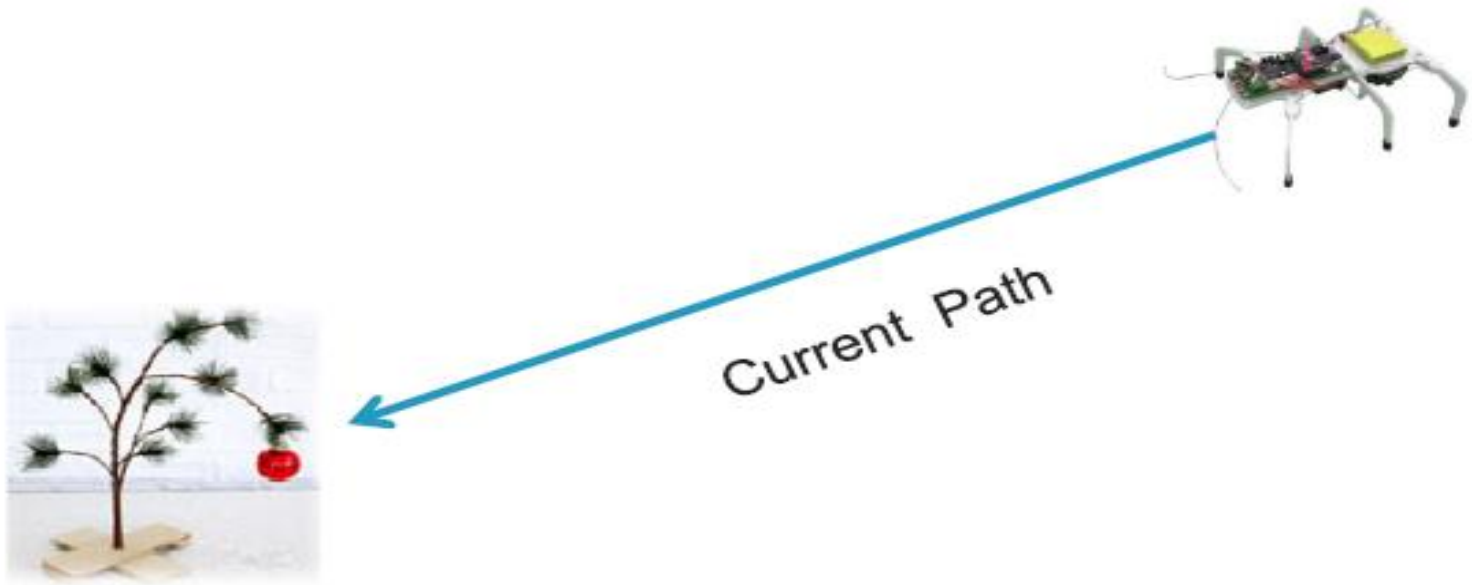
$\Delta e \longrightarrow$

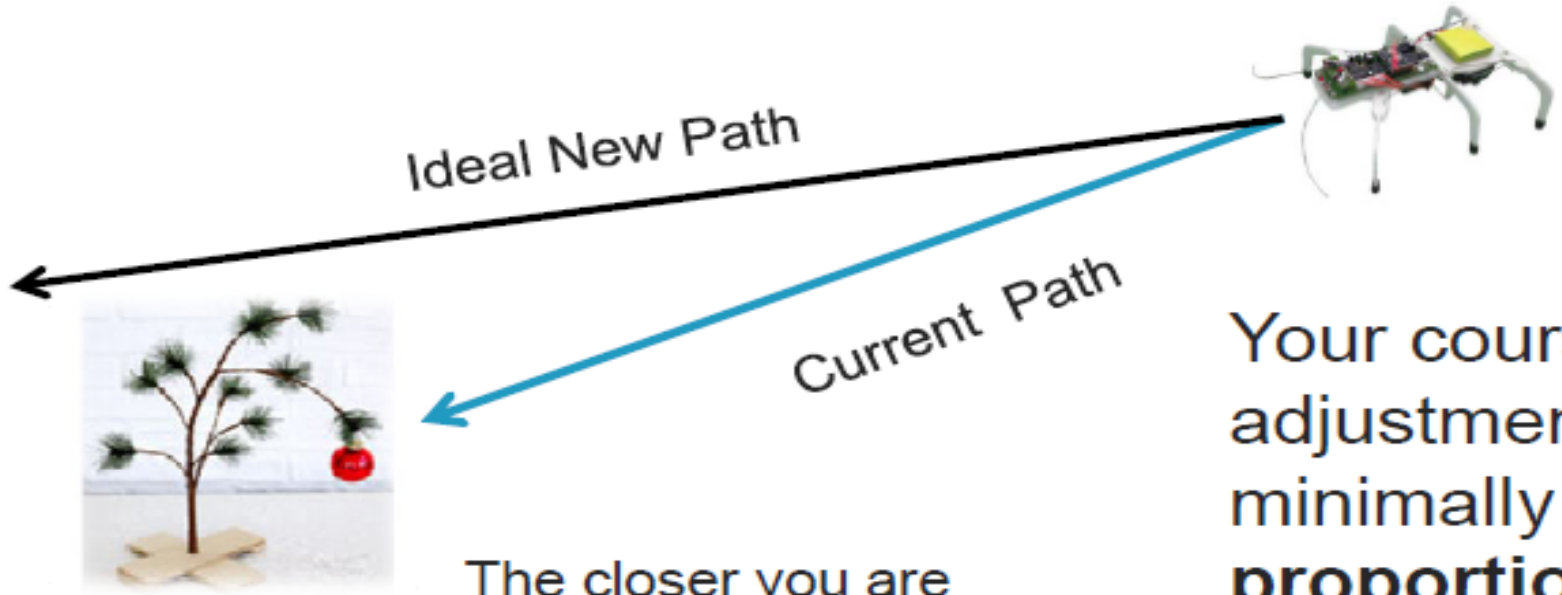
$e \downarrow$

	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NB	NM	NS	ZE
NM	NB	NB	NB	NM	NS	ZE	PS
NS	NB	NB	NM	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PM	PB	PB
PM	NS	ZE	PS	PM	PB	PB	PB
PB	ZE	PS	PM	PB	PB	PB	PB

Problem:

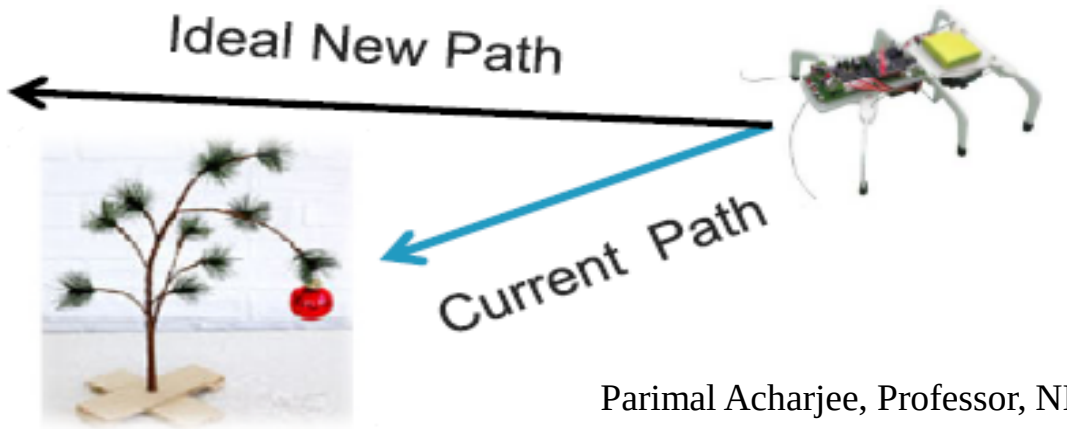
- We have a robot and we want it to move around obstacles based on how close we are to them. How do we do this?





The closer you are to an obstacle, the harder you need to turn to avoid it.

Your course adjustments are minimally **proportional** to the distance to an obstacle and your current speed and heading.



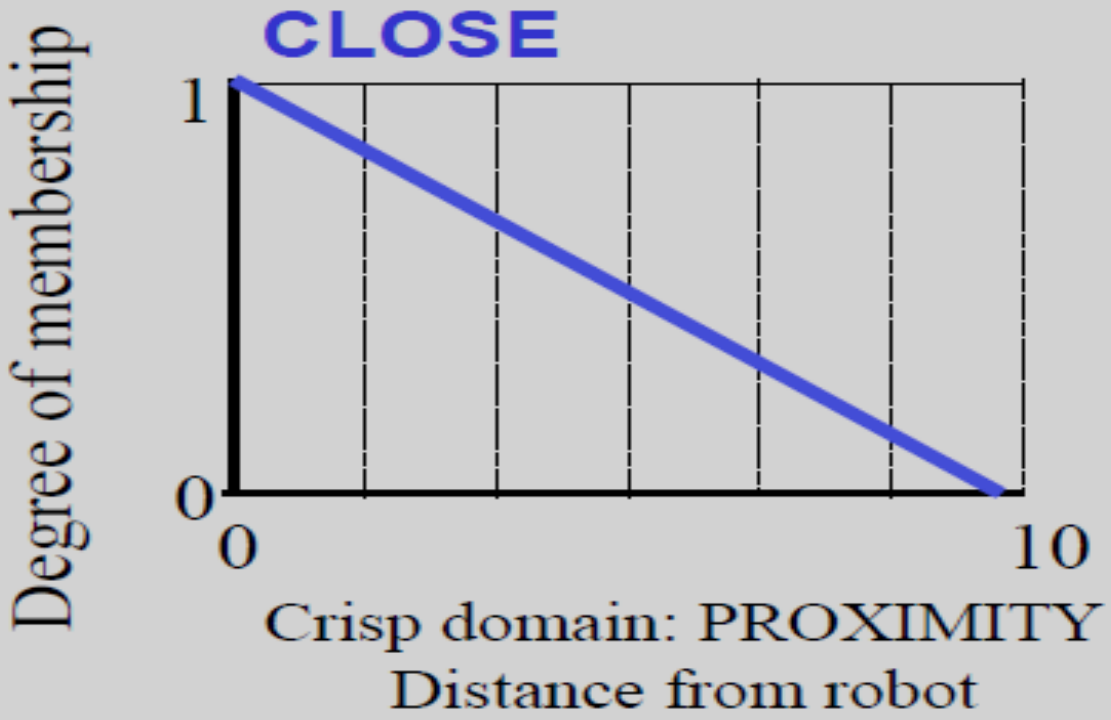
Motivating Example: Swerving Robot

- "Swerve" is a runaway behavior that doesn't let the robot turn more than 90 degrees from current direction
  - Vdir: [0,90]
  - Closer the robot, the harder the turn to the right
- Robot velocity is same (for simplicity)

Linguistic-based rules

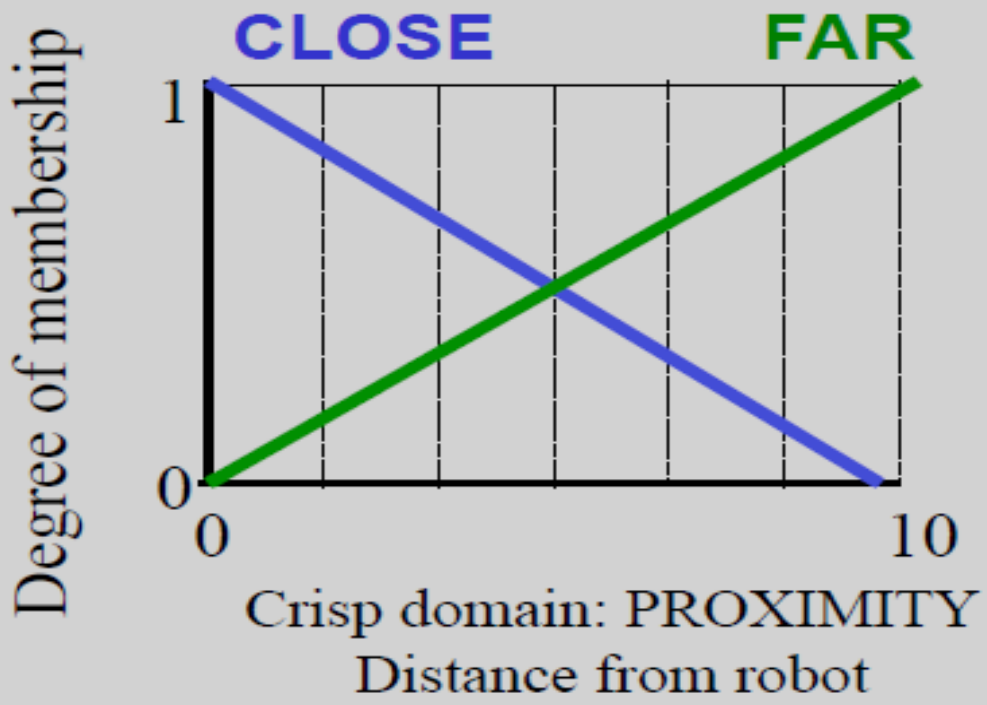
- Swerve 1 (from sensor 1)
  - If an obstacle is close, take a HARD RIGHT
- Swerve 2 (from sensor 2)
  - If an obstacle is far, take a SOFT RIGHT

Fuzzy Sets: Domain



CLOSE is a fuzzy set over the domain

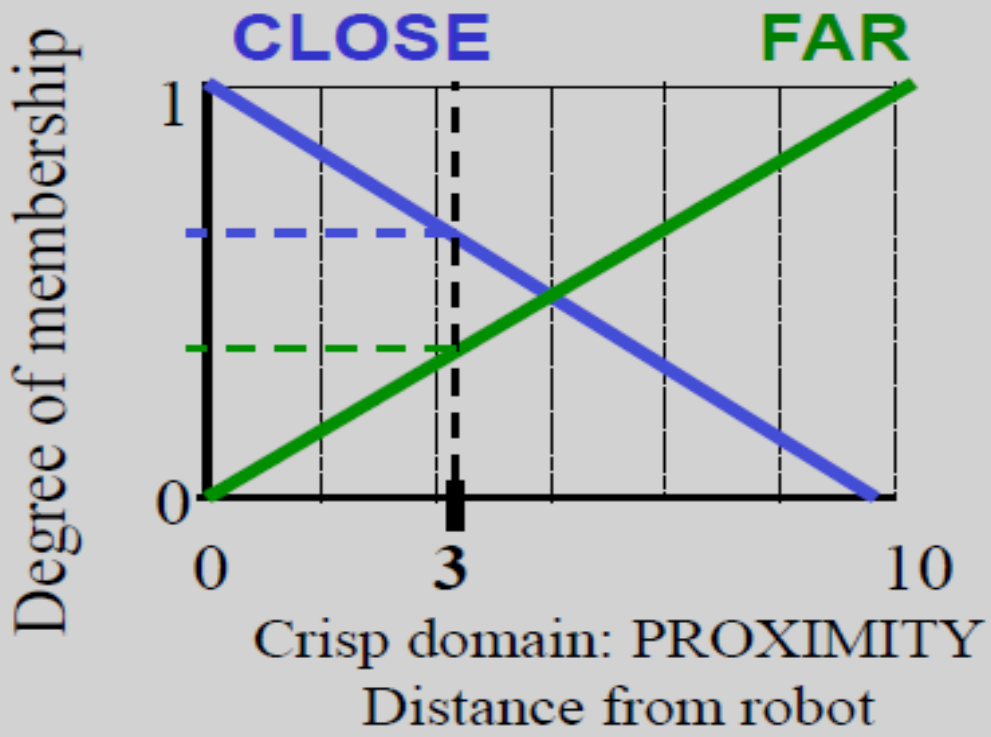
Fuzzy Sets: Multiple Sets



Notes:

- Fuzzy sets often overlap -- that's seen as a Good Thing
- Set can have different shapes (lines, trapezoids, sigmoids)

Membership Functions



If robot is 3 meters from obstacle, it has a membership in CLOSE of 0.7 and a membership in FAR of 0.3

$$M_{CLOSE}(3)=0.7 \quad M_{FAR}(3)=0.3$$

Fuzzy Rules

Motor Schema might be expressed as rule(s):

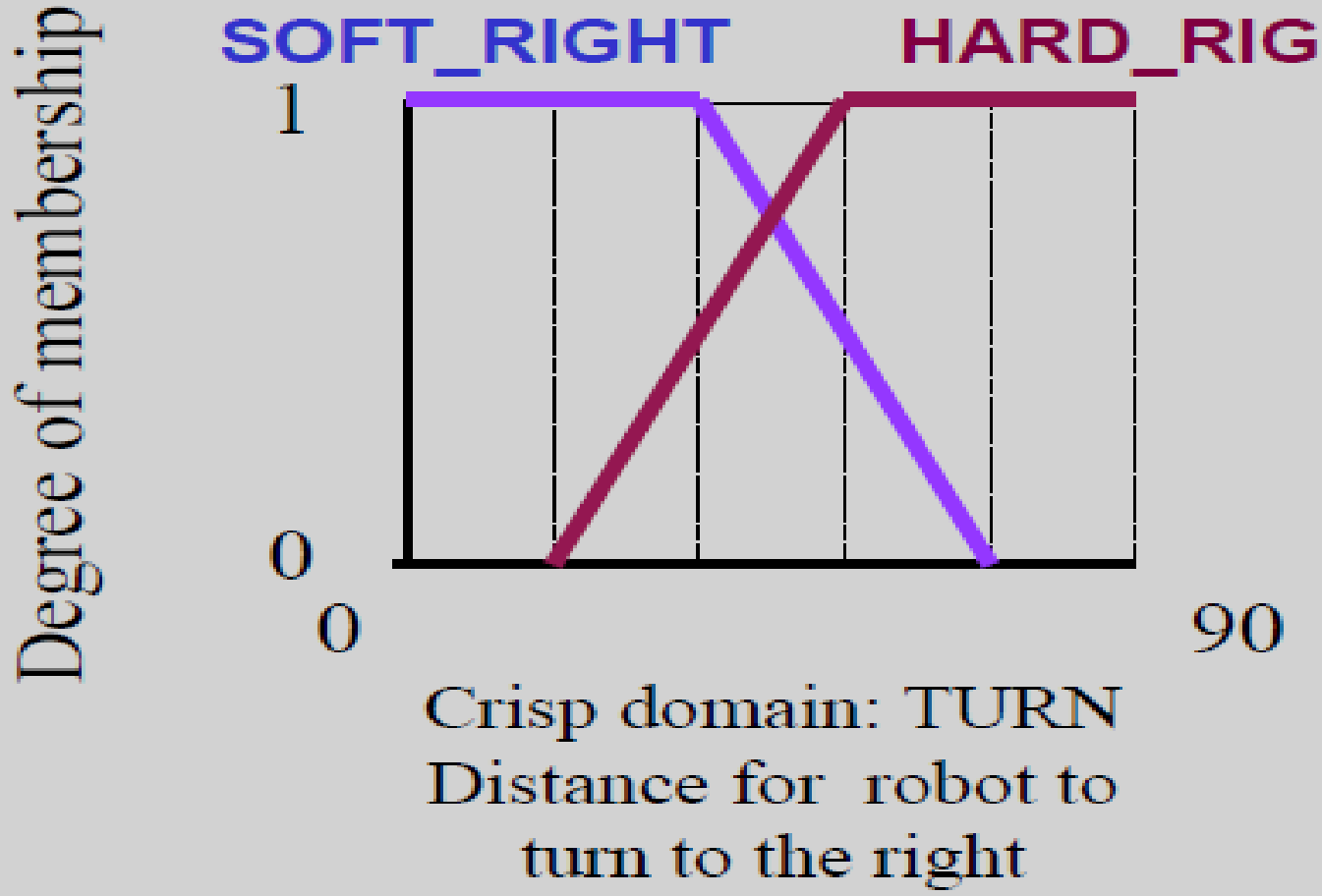
If PROXIMITY is CLOSE
TURN is HARD_RIGHT

Fuzzy rule

If PROXIMITY is FAR
TURN is SOFT_RIGHT

Fuzzy rule

Fuzzy Output Variable



Strength

Motor Schema might be expressed as rule(s):

$M_{CLOSE}(3)=0.7$

If PROXIMITY is CLOSE

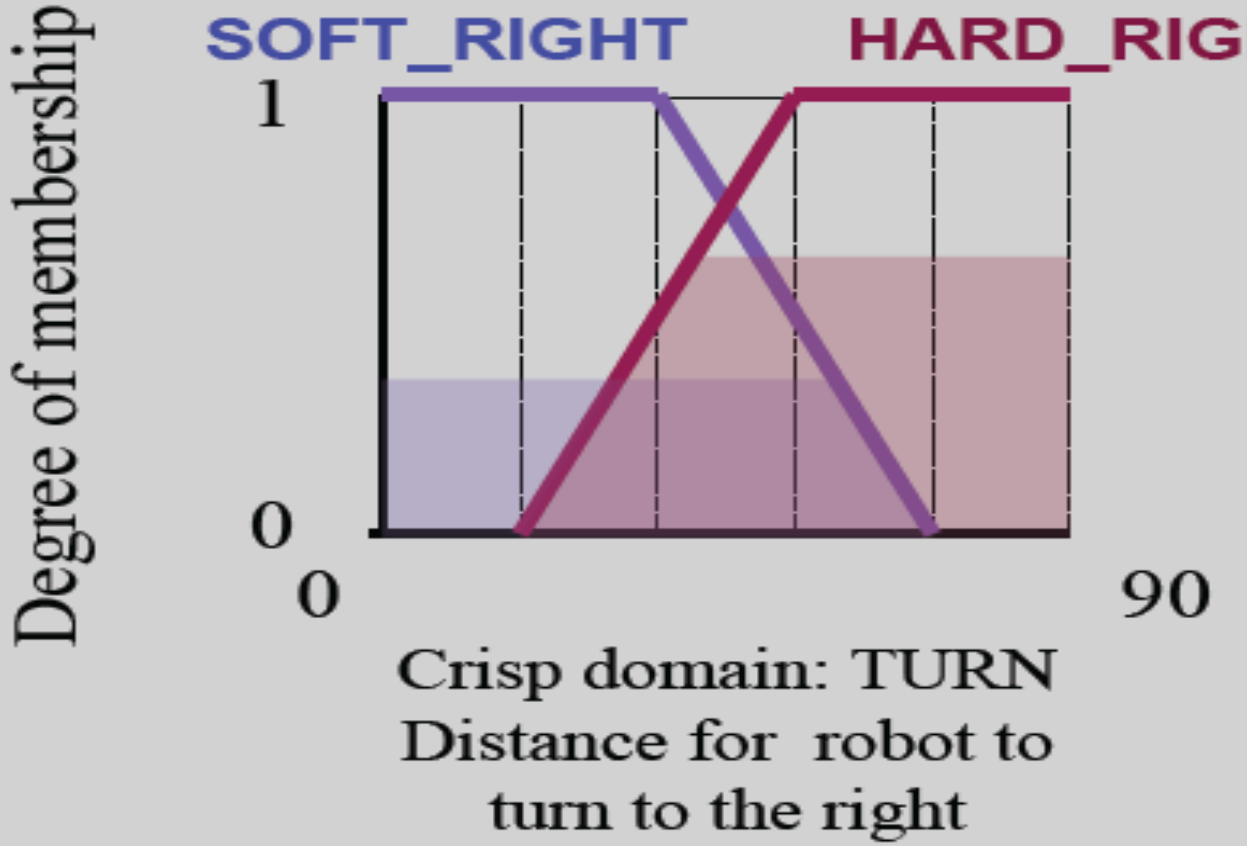
TURN is HARD_RIGHT $M_{HARD\_RIGHT}=0.7$

$M_{FAR}(3)=0.3$

If PROXIMITY is FAR

TURN is SOFT_RIGHT $M_{SOFT\_RIGHT}=0.3$

Resulting Membership



$M_{SOFT\_RIGHT}=0.3$

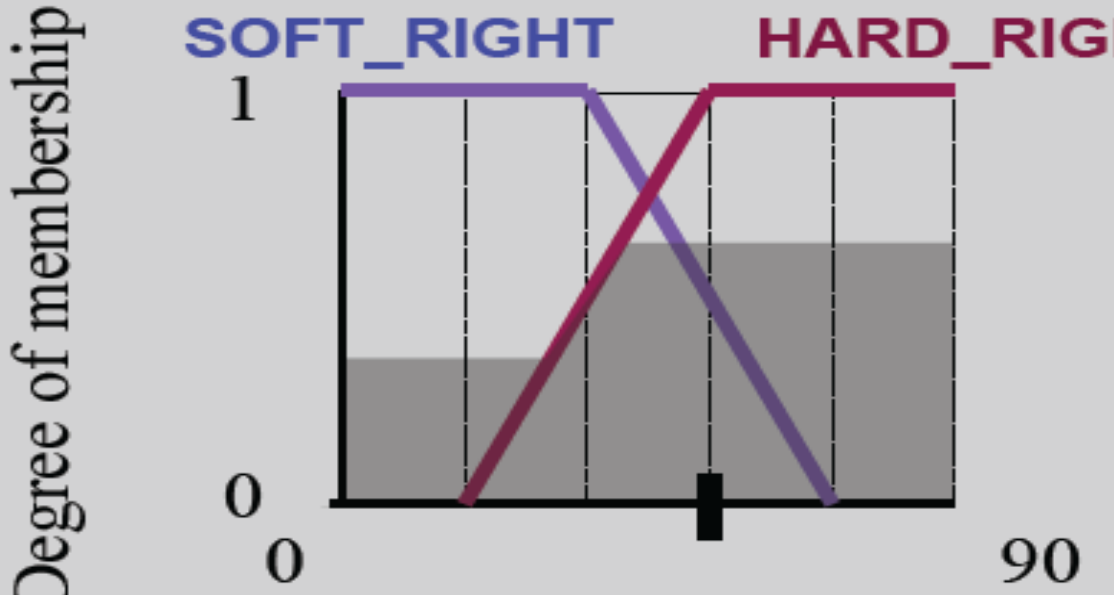
$M_{HARD\_RIGHT}=0.7$

Defuzzification

- Now we have an output that is a fuzzy variable, but we need to convert it to a crisp value to actually send the motor commands.

- Several alternatives:
  - Take Centroid (along Crisp axis) of Blended Area
  - Take Centroid of Largest Area
  - Weighted Means in area of overlap

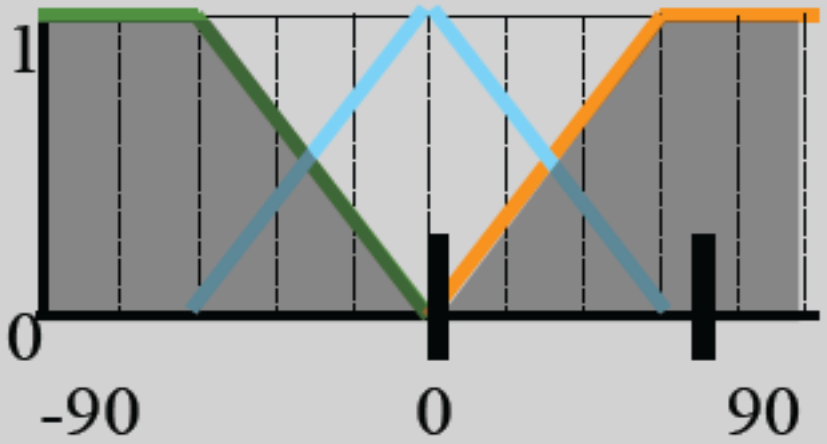
Defuzzification: Blended Centroid



$$C = \frac{\int M_A(X)X}{\int M_A(X)} \qquad C \approx \frac{174.6}{3.2} = 54.6$$

There can be some problems…

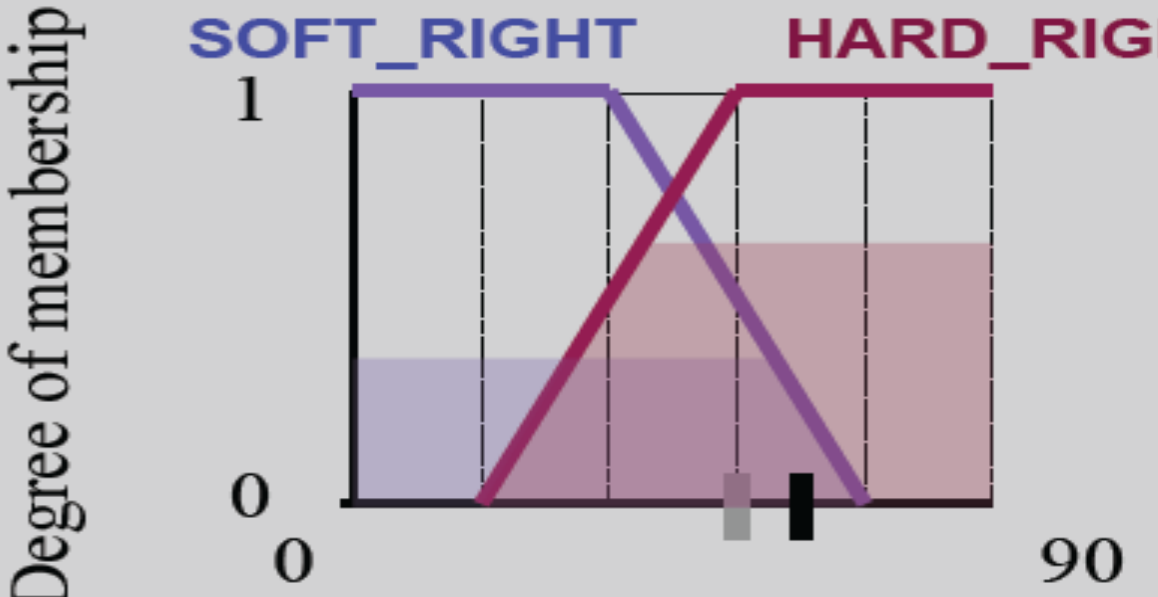
Consider the output of an avoid…



And where would the centroid be?

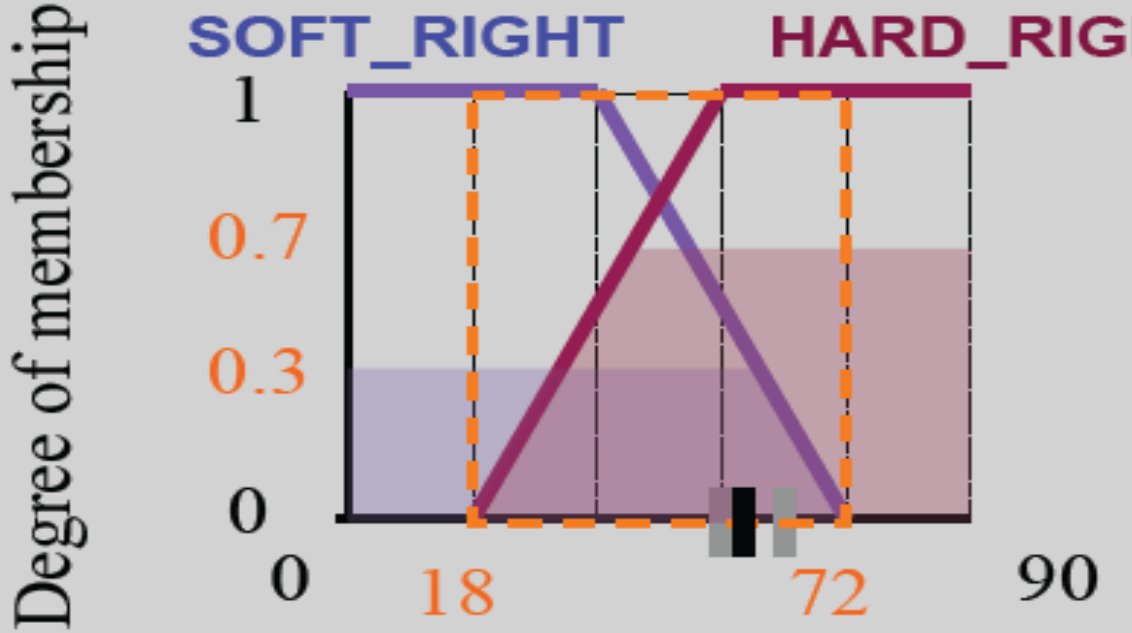
Consider Centroid of Largest Area (CLA)

Defuzzification: Largest Centroid



$$C = \frac{\int M_A(X)X}{\int M_A(X)} \qquad C \approx 61.7$$

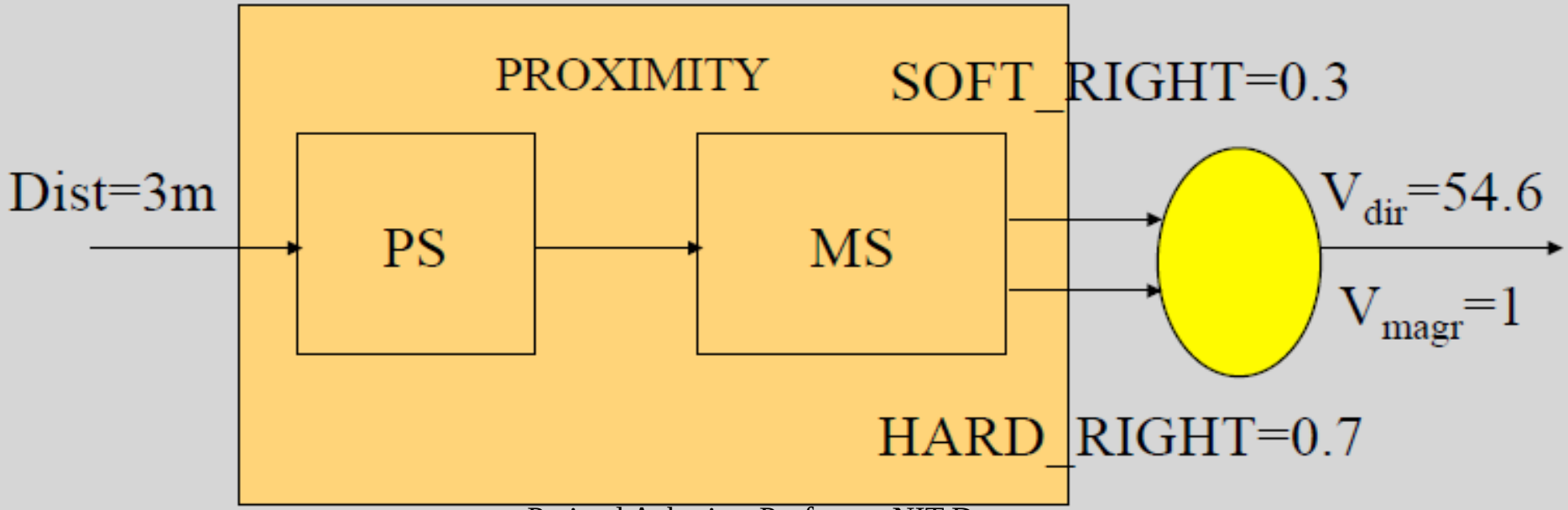
Defuzzification: Weighted Means



$$C = 0.3(18) + 0.7(72) = 55.8$$

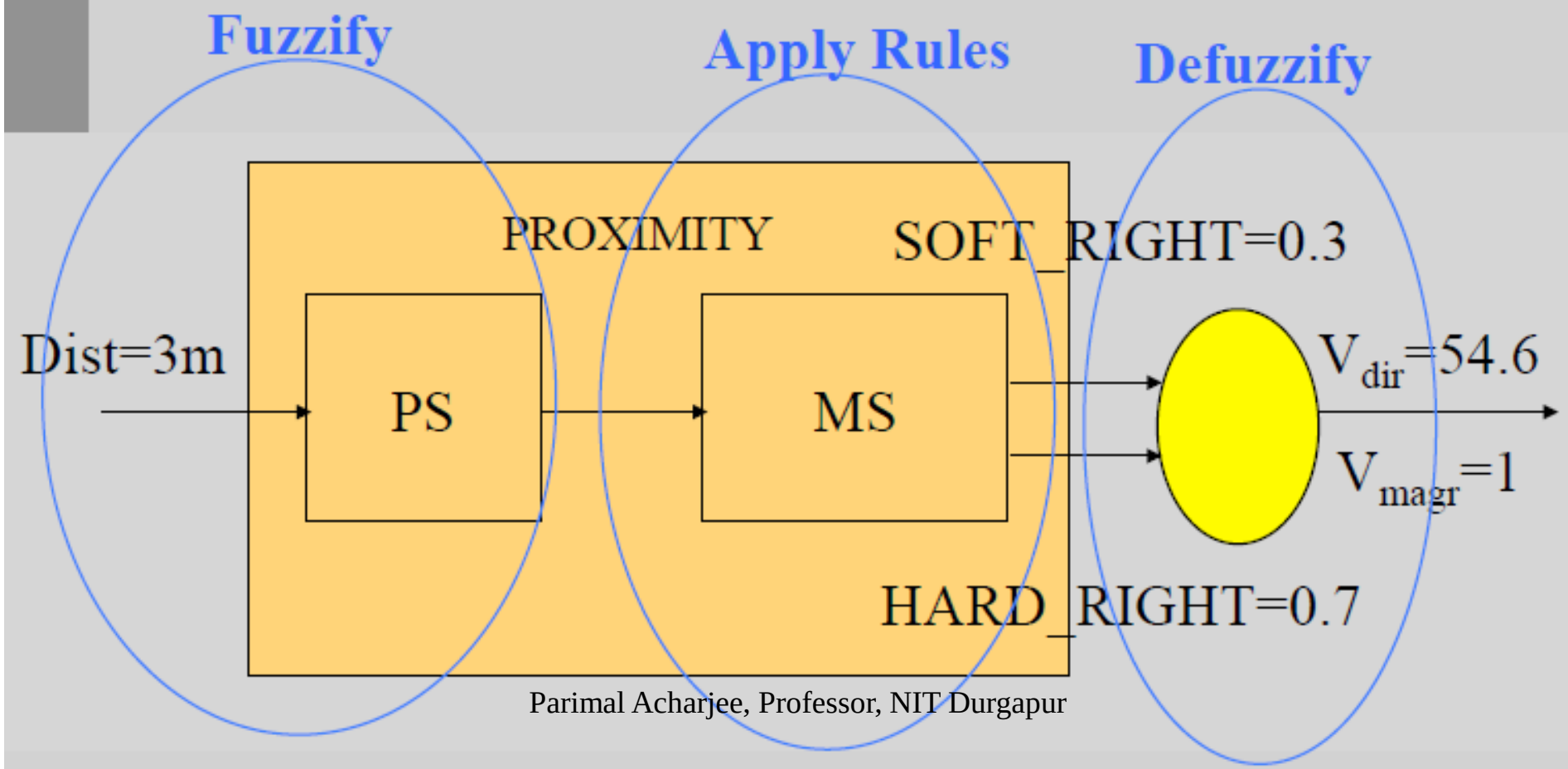
Back to Swerve

- And so we get an answer!



Back to Swerve

- And so we get an answer!



If PROXIMITY is CLOSE AND OPEN is RIGHTSIDE
TURN is HARD_RIGHT

If PROXIMITY is CLOSE AND OPEN is LEFTSIDE
TURN is HARD_LEFT

If PROXIMITY is FAR AND OPEN is RIGHTSIDE
TURN is SOFT_RIGHT

If PROXIMITY is FAR AND OPEN is LEFTSIDE
TURN is SOFT_LEFT

Another set of Rules

Parimal Acharjee, Professor, NIT Durgapur

Summary

- Many packages have fuzzy logic: Java, MATLAB
- Fuzzy works by
  - Fuzzification of crisp values
  - Application of rules, each of which has a strength
  - Defuzzification output of rules to produce a crisp output
- Problems include
  - Results may not be what was expected
  - The number and shape of sets impact behavior

Mamdani-Type Fuzzy Inference

- It is the most commonly seen fuzzy methodology.
- Mamdani's method was among the first control systems built using fuzzy set theory.
- Ebrahim Mamdani developed a new fuzzy inference based on Lotfi Zadeh's 1973 paper.
- It was proposed in 1975 by Ebrahim Mamdani.
- Dr. Ebrahim Mamdani applied it to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators.

Output and Defuzzification

- *Mamdani-type inference* expects the output membership functions to be fuzzy sets.
- After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification.

Takagi-Sugeno-Kang (TKS) method of fuzzy inference

- Introduced in 1985
- similar to the Mamdani method in many respects
- The first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator, are exactly the same.
- The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant.

Output of TSK

A typical rule in a Sugeno fuzzy model has the form:

If Input 1 = x and Input 2 = y, then Output is:

$$z = ax + by + c$$

For a zero-order Sugeno model, the output level z is a constant $(a=b=0)$.

Advantages of the Sugeno Method

- It is computationally efficient.
- It works well with linear techniques (e.g., PID control).
- It works well with optimization and adaptive techniques.
- It has guaranteed continuity of the output surface.
- It is well suited to mathematical analysis.

Advantages of the Mamdani Method

- It is intuitive.
- It has widespread acceptance.
- It is well suited to human input.
- It can be apply with other techniques such as evolutionary computation (EC) and ANN.

Limitations of Fuzzy Controller

- Extensive expertise has to be available.
- Application specific.
- No standard tuning technique.
- No scope for rule refinement.
- Performance degrades with reduced number of rules.

Applications

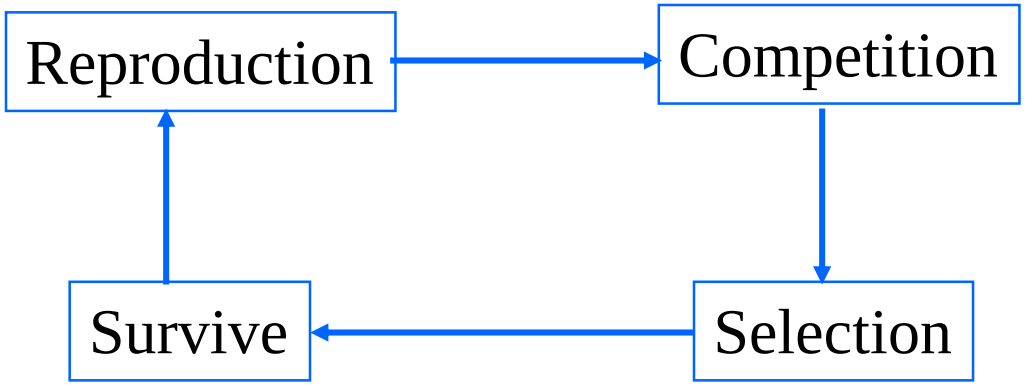
- Expert Systems
- Control Units
- Bullet train between Tokyo and Osaka
- Video Cameras
- Automatic Transmissions
- ABS Brakes
- Refrigerator
- Air Condition
- Washing Machine

Genetic Algorithm Approach

Dr. Parimal Acharjee
Professor
Electrical Engineering
National Institute of Technology
Durgapur

Genetic Algorithm

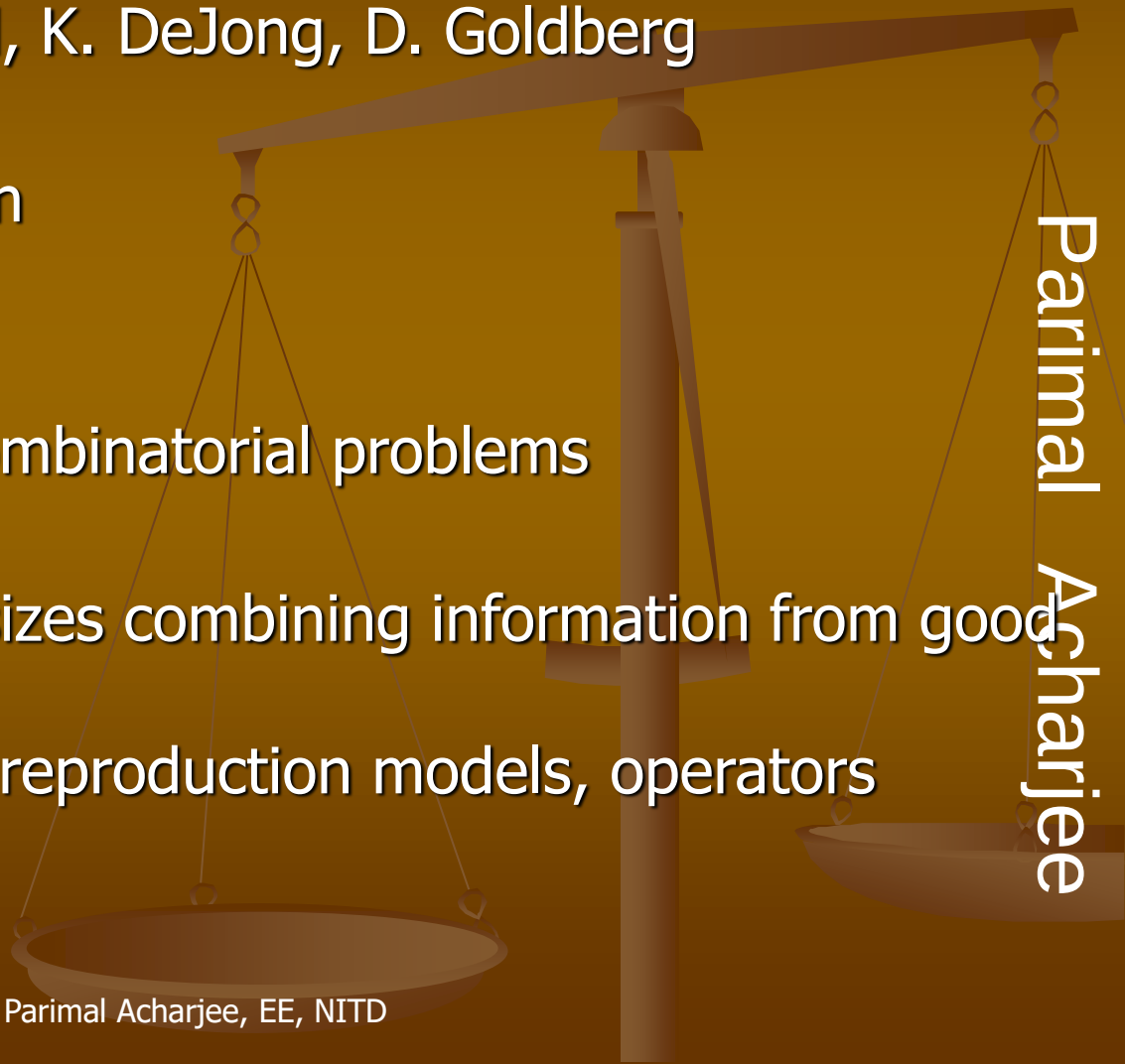
- Based on Darwinian Paradigm



- Intrinsically a robust search and optimization mechanism

GA Quick Overview

- Developed: USA in the 1970's
- Early names: J. Holland, K. DeJong, D. Goldberg
- Typically applied to:
  - discrete optimization
- Attributed features:
  - not too fast
  - good heuristic for combinatorial problems
- Special Features:
  - Traditionally emphasizes combining information from good parents (crossover)
  - many variants, e.g., reproduction models, operators



Genetic Algorithm Introduction

- Inspired by natural evolution
- Population of individuals
  - Individual is feasible solution to problem
- Each individual is characterized by a Fitness function
  - Higher fitness is better solution
- Based on their fitness, parents are selected to reproduce offspring for a new generation
  - Fitter individuals have more chance to reproduce
  - New generation has same size as old generation; old generation dies
- Offspring has combination of properties of two parents
- If well designed, population will converge to optimal solution

Algorithm

1. Start
2. Initialization/Generate Data
3. Loop

   Parents selection

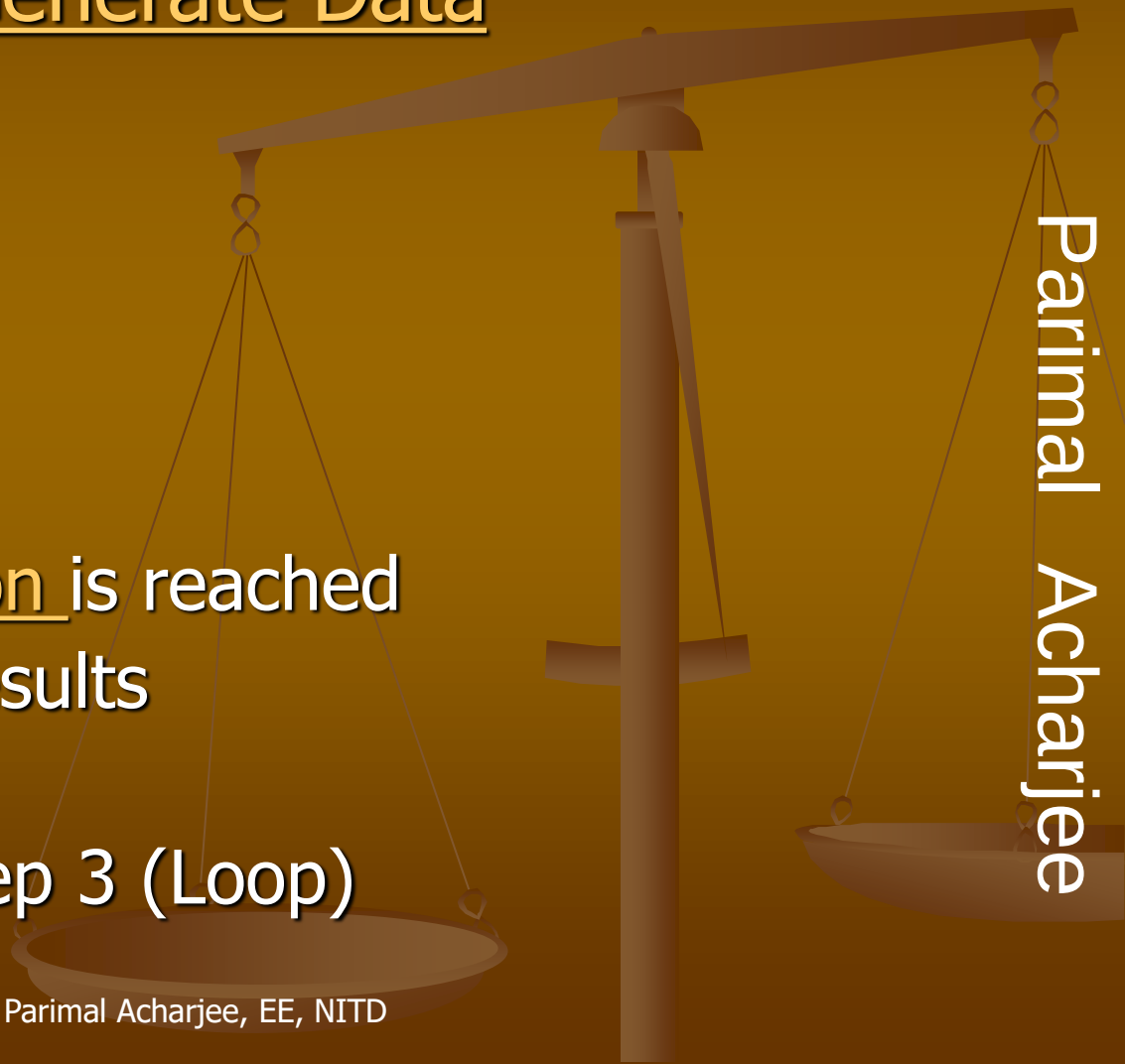
   a) Crossover
   b) Mutation
   c) Judge Fitness
   d) if end condition is reached

      Show Results

      else

      Go to step 3 (Loop)

      end



Basic Principles

- An individual is characterized by a set of parameters: Genes
- Parameters of the solution (genes) are concatenated to form a string (chromosome)
- The chromosome forms the genotype
- The genotype contains all information to construct an organism: the phenotype

- Reproduction is a "dumb" process on the chromosome of the genotype
- Fitness is measured in the real world ('struggle for life') of the phenotype

Genotype

- Genotype can be described as the genetic makeup of an organism.
- The genotype is a set of genes in the DNA which are responsible for the unique trait or characteristics.
- The human genetic code could be found by their genotype. It determines the traits which will be expressed.
- Genotype can be determined by biological tests.
- Examples: Blood group, eye colour, height, genetic diseases.

Phenotype

- The phenotype is the physical appearance or characteristic of the organism.
- A key difference between phenotype and genotype is that, while genotype is inherited from the parents but the phenotype is not.
- Phenotype depends on the genotype but also influenced by the factors such as environment (nutrition, temperature, humidity, and stress), life style, food habit.
- Examples: Weight, physique, beak of birds, skin color.

Examples

- Our genes can control the amount and type of melanin produced by us. But, exposure to UV light in sunny climates causes the darkening of existing melanin and hence it caused darker skin.
- Flamingos are another suitable example of how the environment influences the phenotype. As they are renowned for being vibrantly pink, but their natural color is white and the pink color is caused by pigments in the organisms in their diet.

Genotype and Phenotype

Genotype	Phenotype
The hereditary information of the organism in the form of gene in the DNA and remains the same throughout the life.	The characters of an organism which are visible are known as phenotypes.
Same genotype produces same phenotype.	Same phenotype may or may not belong to same genotype.
Present inside the body as genetic material.	Expression of genes as the external appearence.
The genotype is inherited from the parent to the offspring.	The phenotype is not inherited from the parent.
It can be determined by scientific methods such as the polymerase chain reaction.	It can be determined by observing the organism.
It is affected by genes.	It is affected by genotype and environmental conditions.
For eg., Blood group, eye colour, height, genetic diseases.	For eg., Weight, physique, beak of birds

Reproduction

- Reproduction operators
  - Crossover
  - Mutation

Reproduction

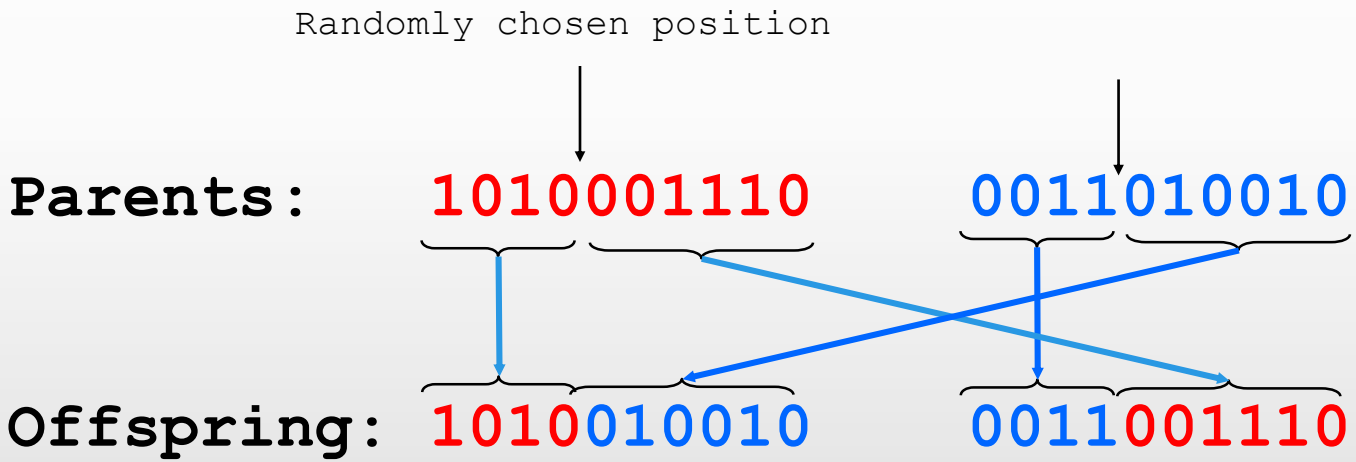
- Crossover
  - Two parents produce two offspring
  - There is a chance that the chromosomes of the two parents are copied unmodified as offspring
  - There is a chance that the chromosomes of the two parents are randomly recombined (crossover) to form offspring
  - Generally the chance of crossover is between 0.6 and 1.0
- Mutation
  - There is a chance that a gene of a child is changed randomly
  - Generally the chance of mutation is low (e.g. 0.001)

Reproduction Operators

- Crossover
  - Generating offspring from two selected parents
    - Single point crossover
    - Two point crossover (Multi point crossover)
    - Uniform crossover

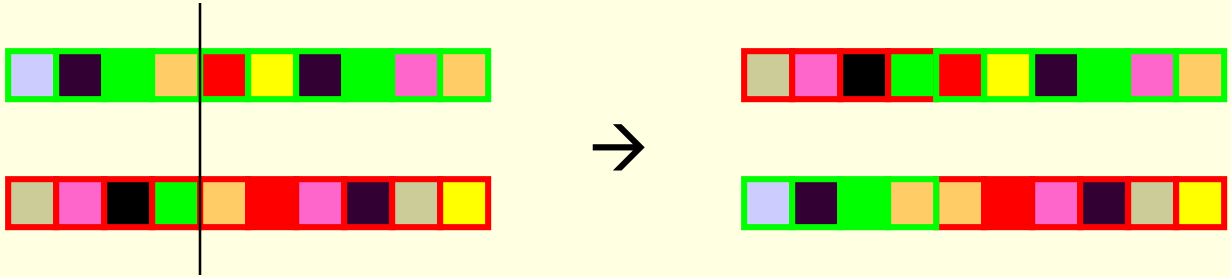
One-point crossover

- Randomly one position in the chromosomes is chosen
- Child 1 is head of chromosome of parent 1 with tail of chromosome of parent 2
- Child 2 is head of 2 with tail of 1



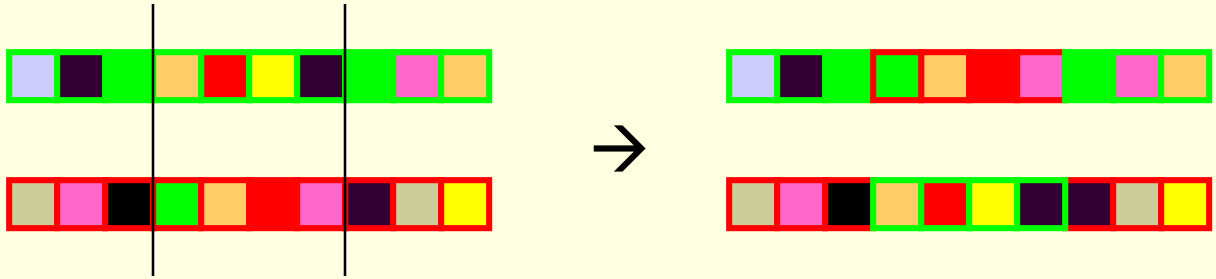
Reproduction Operators comparison

- Single point crossover

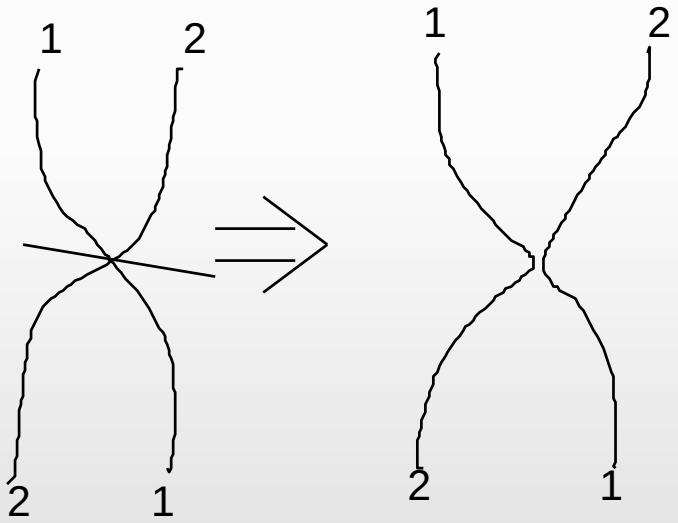


Cross point

- Two point crossover (Multi point crossover)

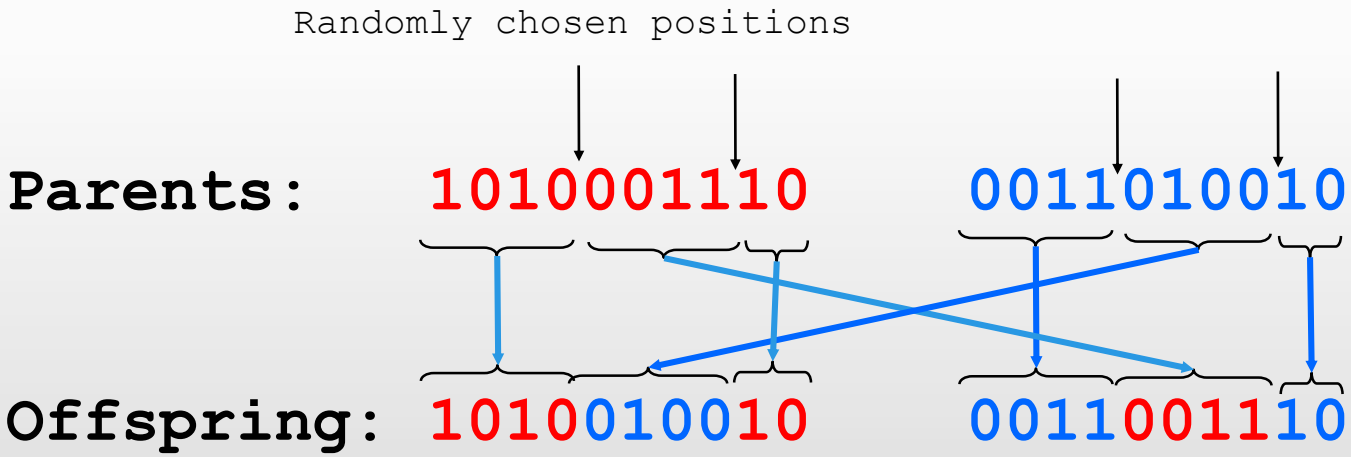


One-point crossover - Nature



Two-point crossover

- Randomly two positions in the chromosomes are chosen
- Check that genes at the head and genes at the tail of a chromosome are always split when recombined



Uniform crossover

- A random mask is generated
- The mask determines which bits are copied from one parent and which from the other parent
- Bit density in mask determines how much material is taken from the other parent (takeover parameter)

Mask:	**0110011000**	**(Randomly generated)**
Parents:	**1010001110**	**0011010010**
Offspring:	**0011001010**	**1010010110**

Reproduction Operators

- Mutation
  - Generating new offspring from single parent

  - Maintaining the diversity of the individuals
    - ☒Crossover can only explore the combinations of the current gene pool
    - ☒Mutation can “generate” new genes

Control parameters

- Control parameters: **population size, crossover/mutation probability**
  - Problem specific
  - Increase population size
    - Increase diversity and computation time for each generation
  - Increase crossover probability
    - Increase the opportunity for recombination but also disruption of good combination
  - Increase mutation probability
    - Closer to randomly search
    - Help to introduce new gene or reintroduce the lost gene

Parent selection

Chance to be selected as parent proportional to fitness

- Roulette wheel

To avoid problems with fitness function

- Tournament

Not a very important parameter

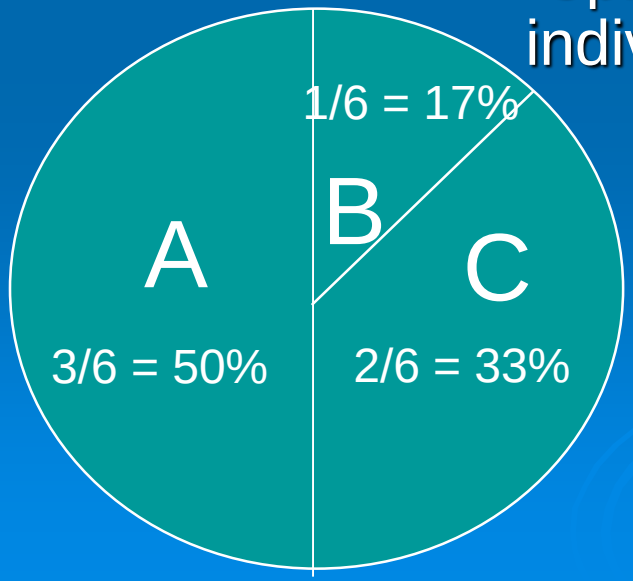
Parimal Acharjee, EE, NITD

GA Parent Selection - Tournament

- Select a pair of individuals at random. Generate a random number, *R*, between 0 and 1. If *R* < *r* use the first individual as a parent. If the *R* >= *r* then use the second individual as the parent. This is repeated to select the second parent. The value of *r* is a parameter to this method
- Select two individuals at random. The individual with the highest evaluation becomes the parent. Repeat to find a second parent.

GA operators: Selection

- Main idea: better individuals get higher chance
  - Chances proportional to fitness
  - Implementation: roulette wheel technique
    - Assign to each individual a part of the roulette wheel
    - Spin the wheel n times to select n individuals



fitness(A) = 3

fitness(B) = 1

fitness(C) = 2

An example

- Simple problem: max x^2 over {0,1,…,31}
- GA approach:
  - Representation: binary code, e.g. 01101 $\leftrightarrow$ 13
  - Population size: 4
  - 1-point xover, bitwise mutation
  - Roulette wheel selection
  - Random initialisation
- We show one generational cycle done by hand

x^2 example: selection

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

X^2 example: crossover

String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

X^2 example: mutation

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	28	784
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	20	400
Sum				2538
Average				634.5
Max				729

Parent/Survivor Selection

- Strategies
  - Survivor selection
    - ☒Always keep the best one
    - ☒Elitist: deletion of the K worst
    - ☒Probability selection : inverse to their fitness
    - ☒Etc.

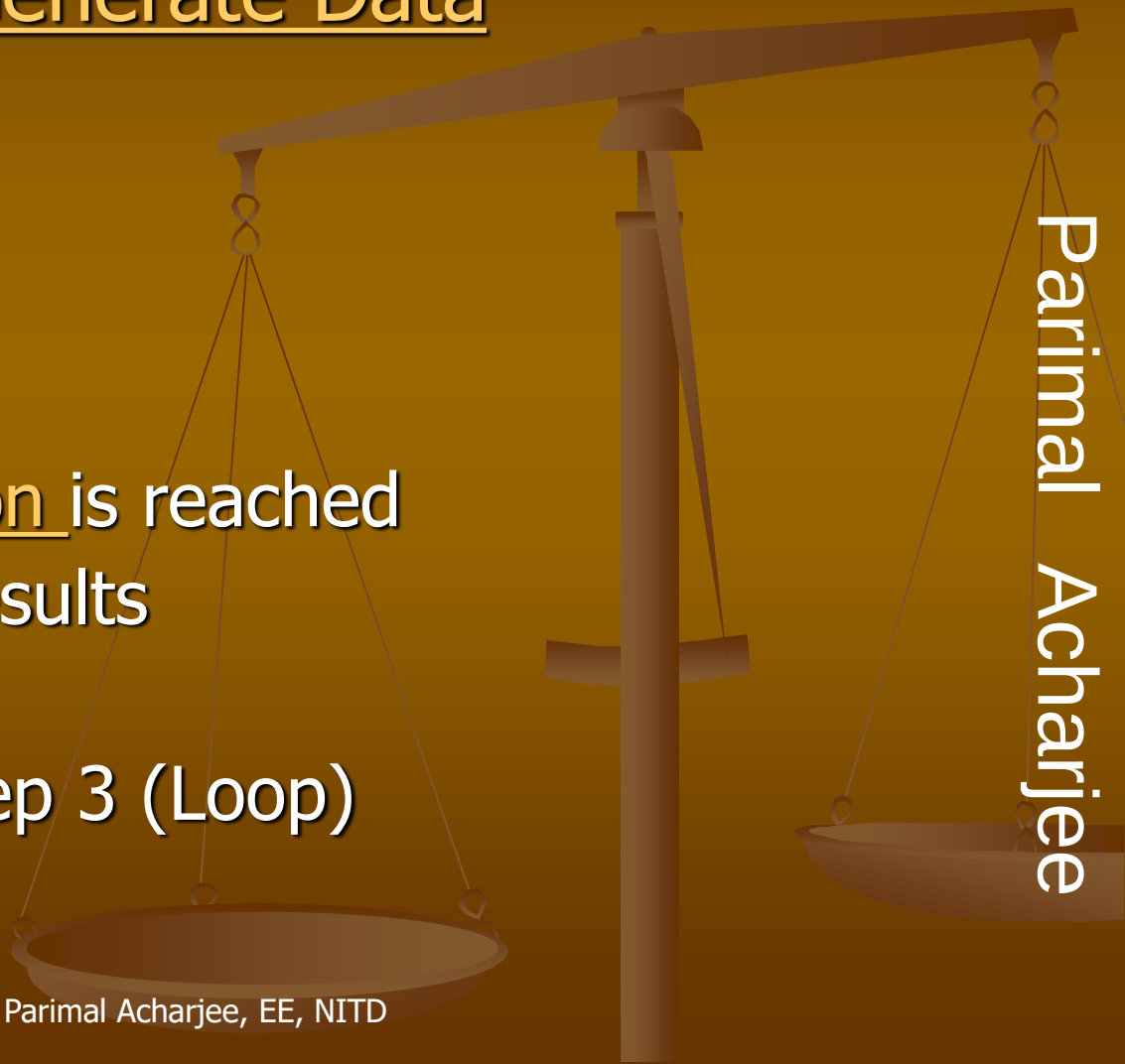
Parent/Survivor Selection

- Too strong fitness selection bias can lead to sub-optimal solution
- Too little fitness bias selection results in unfocused and meandering search

Parimal Acharjee, EE, NITD

Algorithm

1. Start
2. Initialization/Generate Data
3. Loop
   a) Crossover
   b) Mutation
   c) Judge Fitness
   d) if end condition is reached
      Show Results
      else
      Go to step 3 (Loop)
      end



Start

```
% Simple GA Algorithm:
% Problem or Objective Function : "f(x)=2*x1^2-3*x1-
    x2^2+4*x2-9"
clc
clear all
x1_lower=input('enter the lower limit of variables x1= ');
x1_upper=input('enter the upper limit of variables x1= ');
x2_lower=input('enter the lower limit of variables x2= ');
x2_upper=input('enter the upper limit of variables x2= ');
pop=input('enter the no of population=');
max_iteration=input('enter the maximum number of
    iteration=');
```

Back

Initialization/Generate Data

```
%% Generate initial data
for i=1:pop
    x(i,1)=unifrnd(x1_lower,x1_upper,1,1);
    x(i,2)=unifrnd(x2_lower,x2_upper,1,1);
End
```

OR

```
X(:,1)=unifrnd(x1_lower,x1_upper,pop,1);
X(:,2)=unifrnd(x2_lower,x2_upper,pop,1);
cp=0.9;                  % Crossover Probability
mp=0.01;                 % Mutation Probability
q=quantizer([20 16]);   % For binary Format
```

Back

Loop

```
for iteration=1:max_iteration
    %Convert number to binery
    [bin_x1,bin_x2]=num2bin(q,x(:,1),x(:,2));
    crossover
    mutation
    judge_fitness
    % Check end condition
    [min_value, indices]=min(abs(fitness))
    if min_value<=0.01
        results=x(indices,:)
        break
    end
    iteration=iteration+1
end
```

Back

Judge fittness

```
[x1,x2]=bin2num(q,child_x1,child_x2);
    x=[x1 x2];
    %Fitness function
    for i=1:pop
        fitness(i)=2*x(i,1)^2-3*x(i,1)-
   x(i,2)^2+4*x(i,2)-9;
    end
```

Back

Crossover

```
offspring_x1=bin_x1;
offspring_x2=bin_x2;
for i=1:2:pop
    cr=unifrnd(0,1);  OR rand()
    if cr<cp
        two_point_crossover
    end
end
```

Back

Mutation

```
child_x1=offspring_x1;
child_x2=offspring_x2;
for i=1:pop
    mr=unifrnd(0,1);
    if mr<mp
        compliment
    end
end
```

Back

Compliment Mutation

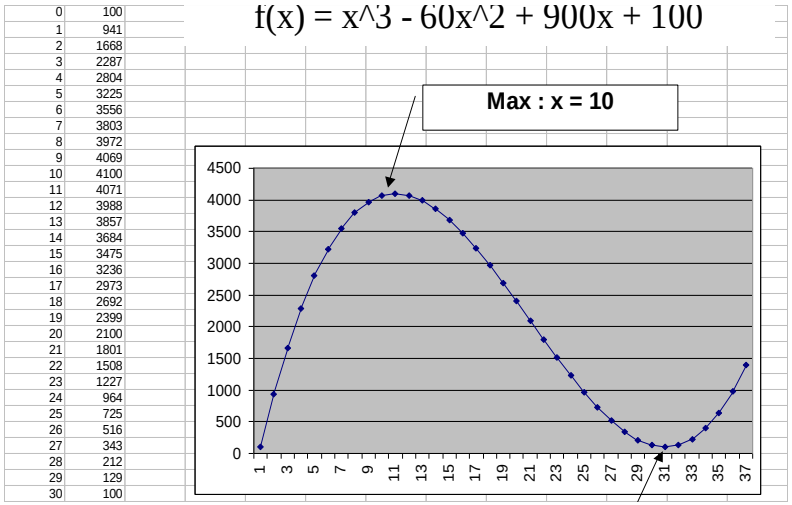
```
mut_x1=bitcmp(bin2dec(child_x1(i,:)),20);

mut_x2=bitcmp(bin2dec(child_x2(i,:)),20);

    child_x1(i,:)=num2bin(q,mut_x1);

    child_x2(i,:)=num2bin(q,mut_x2);
```

Back

GA Example

- **Crossover probability, $P_C = 1.0$**
- **Mutation probability, $P_M = 0.0$**
- **Maximise $f(x) = x^3 - 60 * x^2 + 900 * x + 100$**
- **$0 <= x <= 31$**
- **x can be represented using five binary digits**

0	100
1	941
2	1668
3	2287
4	2804
5	3225
6	3556
7	3803
8	3972
9	4069
10	4100
11	4071
12	3988
13	3857
14	3684
15	3475
16	3236
17	2973
18	2692
19	2399
20	2100
21	1801
22	1508
23	1227
24	964
25	725
26	516
27	343
28	212
29	129
30	100



GA Example

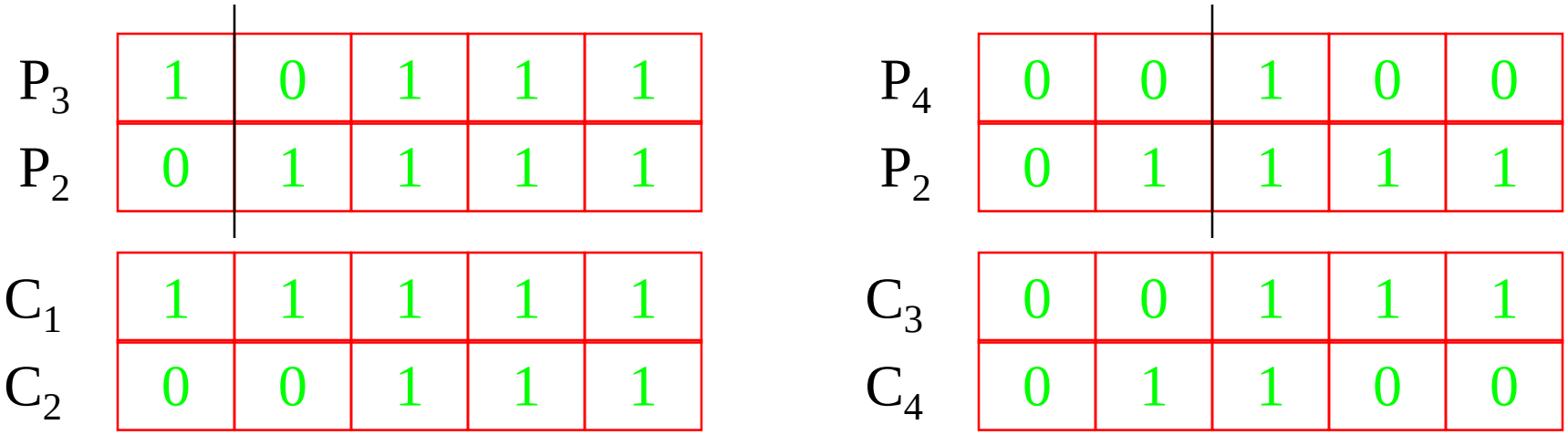
- **Generate random individuals**

Chromosome	Binary String	x	f(x)
P_1	11100	28	212
P_2	01111	15	3475
P_3	10111	23	1227
P_4	00100	4	2804
	TOTAL		**7718**
	AVERAGE		**1929.50**

GA Example

- **Choose Parents, using roulette wheel selection**
- **Parents P_3 & P_2 and P_4 & P_2 are selected using roulette wheel selection**
- **Crossover point is chosen randomly**
- **For first case, position of crossover point=1 and For next case, position of crossover point=2**

GA Example - Crossover



GA Example - After First Round of Breeding

- **The average evaluation has risen**
- **P_2, was the strongest individual in the initial population. It was chosen both times but we have lost it from the current population**
- **We have a value of x=7 in the population which is the closest value to 10 we have found**

Chromosome	Binary String	x	f(x)
P_1	11111	31	131
P_2	00111	7	3803
P_3	00111	7	3803
P_4	01100	12	3988
	TOTAL		**11725**
	AVERAGE		**2931.25**

Real Coded Genetic Algorithm (RCGA)

- No requirement of conversion i.e. number to binary & binary to number.
- Reproduction is completed with real value i.e. crossover and mutation are implemented with real data (not binary).
- RCGA is easy to implement.
- It may give sub-optimal solution or suffer from convergence problem.
- Now-a-days, RCGA is used for different types of engineering problems (complicated complex problems)

CROSSOVER

- Arithmetic crossover
- Heuristic crossover

ARITHMETIC CROSSOVER

- Linearly combines two parent chromosome vectors to produce two new offspring

$$Offspring1 = a * Parent1 + (1- a) * Parent2$$
$$Offspring2 = (1 – a) * Parent1 + a * Parent2$$

where a is a random weighting factor (chosen before each crossover operation).

ARITHMETIC CROSSOVER

2 parents (each consisting of 4 float genes) are (randomly) selected for crossover:

Parent 1: (0.3) (1.4) (0.2) (7.4)
Parent 2: (0.5) (4.5) (0.1) (5.6)

If a = 0.7, the following two offspring would be produced:
Offspring1 = a * *Parent1* + (1- a) * *Parent2*
Offspring2 = (1 – a) * *Parent1* + a * *Parent2*

Offspring1 = 0.7 * *0.3* + 0.3* *0.5=0.36*
Offspring2 = 0.3* *0.3* + 0.7 * *0.5=0.44*

Offspring1: (0.36) (2.33) (0.17) (6.86)
Offspring2: (0.44) (3.57) (0.13) (6.14)

Heuristic crossover

- A crossover operator that uses the fitness values of the two parent chromosomes to determine the direction of the search.
- According to fitness, determine best and worst parent from two selected parents.
- Generate r which is a random number between 0 and 1.

Offspring1 = BestParent + r * (*BestParent – WorstParent*)
Offspring2 = BestParent

SINGLE POINT CROSSOVER

2 parents (each consisting of 4 float genes) are (randomly) selected for crossover:

Parent 1: (0.3) (1.4) (0.2) (7.4)
Parent 2: (0.5) (4.5) (0.1) (5.6)

Random position selected for crossover is '2'.

Parent 1: (0.3) (1.4) | (0.2) (7.4)
Parent 2: (0.5) (4.5) | (0.1) (5.6)

After crossover:

Offspring 1: (0.3) (1.4) (0.1) (5.6)
Offspring 2: (0.5) (4.5) (0.2) (7.4)

Parimal Acharjee, EE, NITD

Real Coded Mutation

- **Gaussian -** A mutation operator that adds a unit Gaussian distributed random value to the chosen gene. The new gene value is clipped if it falls outside of the user-specified lower or upper bounds for that gene. This mutation operator can only be used for integer and float genes.
-

Mutation (Real-Coded)

General Real coded mutation:

$$x(i) = x(i) + N(0, \sigma_i)$$

N(0, σi) is an independent random Gaussian number with the mean of zero and the standard deviation of σi.

Advanced Real coded mutation:

$$x(i) = x(i) + N(0, |x_u - x_l|)$$

Or

$$x(i) = x(i) + N(0, |x_u - x_l|/div)$$

'*div*' is the user-defined division factor.


THANK
YOU

PARTICLE SWARM OPTIMIZATION

DR. PARIMAL ACHARJEE

PROFESSOR

ELECTRICAL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY DURGAPUR

Parimal Acharjee

SOCIAL BEHAVIOR OF BIRDS AND FISHES

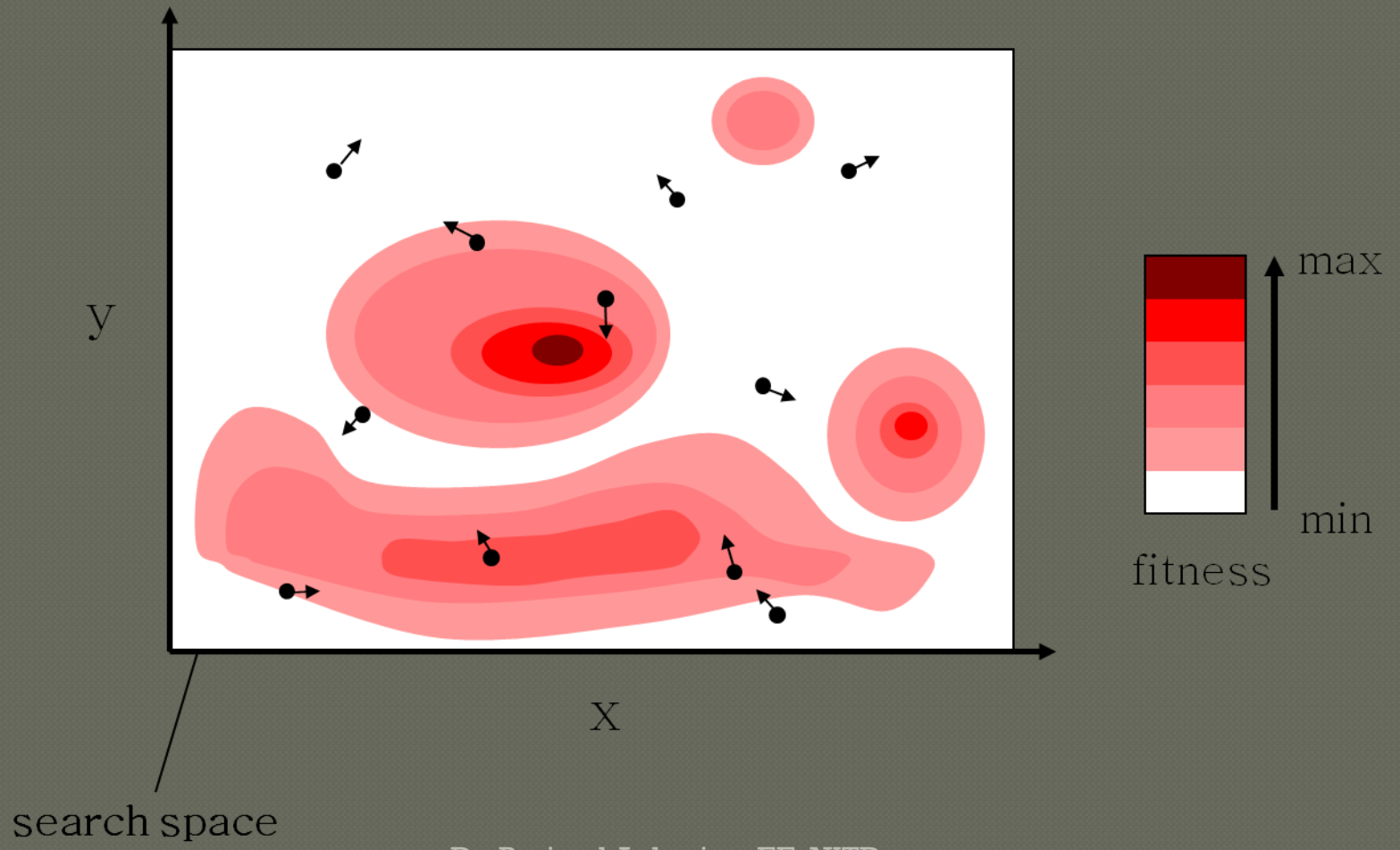
- PARTICLE SWARM OPTIMIZATION(PSO)
  - PROPOSED BY JAMES KENNEDY & RUSSELL EBERHART IN 1995
  - INSPIRED BY SOCIAL BEHAVIOR OF BIRDS AND FISHES
  - COMBINES SELF-EXPERIENCE WITH SOCIAL EXPERIENCE
  - POPULATION-BASED OPTIMIZATION

BIRD FLOCKING

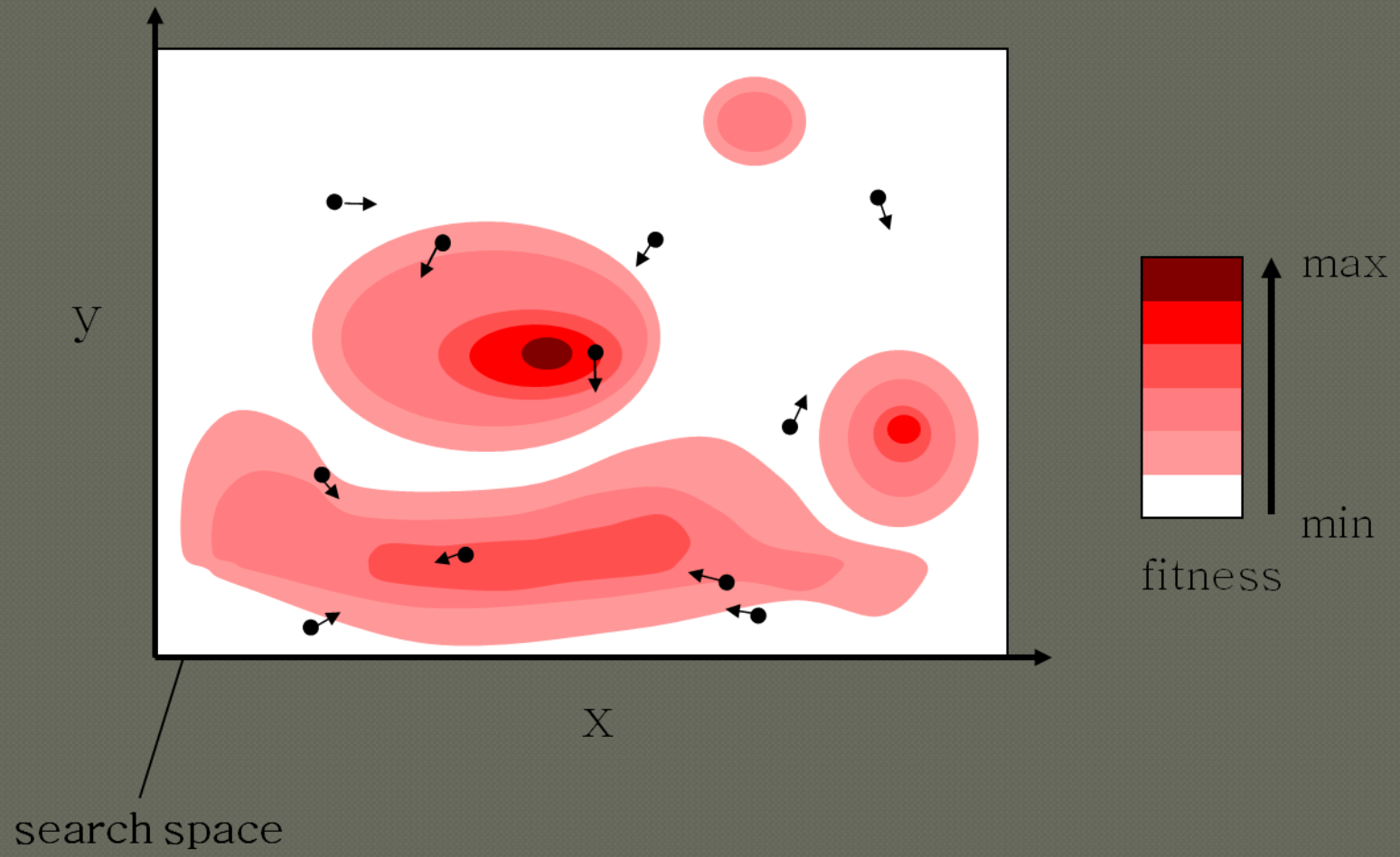
- BIRD FLOCKING STATES HOW BIRDS SEARCH FOOD. ONLY ONE PIECE OF FOOD IS AVAILABLE IN AN AREA.
- A GROUP OF BIRDS ARE RANDOMLY SEARCHING THE FOOD IN THAT AREA.
- THOUGH ALL THE BIRDS DO NOT KNOW WHERE THE FOOD IS, BUT THEY KNOW HOW FAR THE FOOD IS. SO THE BEST WAY TO SEARCH THE FOOD IS THE EFFECTIVE ONE SHOULD FOLLOW THE BIRD THAT IS NEAREST TO THE FOOD.
- PSO SIMULATES THE BEHAVIOURS OF BIRD FLOCKING. TO SOLVE THE OPTIMIZATION PROBLEMS, THIS TECHNIQUE IS USED.
- IN SEARCH SPACE, EACH SOLUTION, "BIRD", IS KNOWN AS PARTICLE.
- USING FITNESS FUNCTION THE FITNESS VALUES OF ALL THE PARTICLES ARE EVALUATED. ALL OF PARTICLES HAVE VELOCITIES THAT DIRECT TO THE CURRENT OPTIMUM PARTICLES AND UPDATE THEIR POSITIONS.

BACK

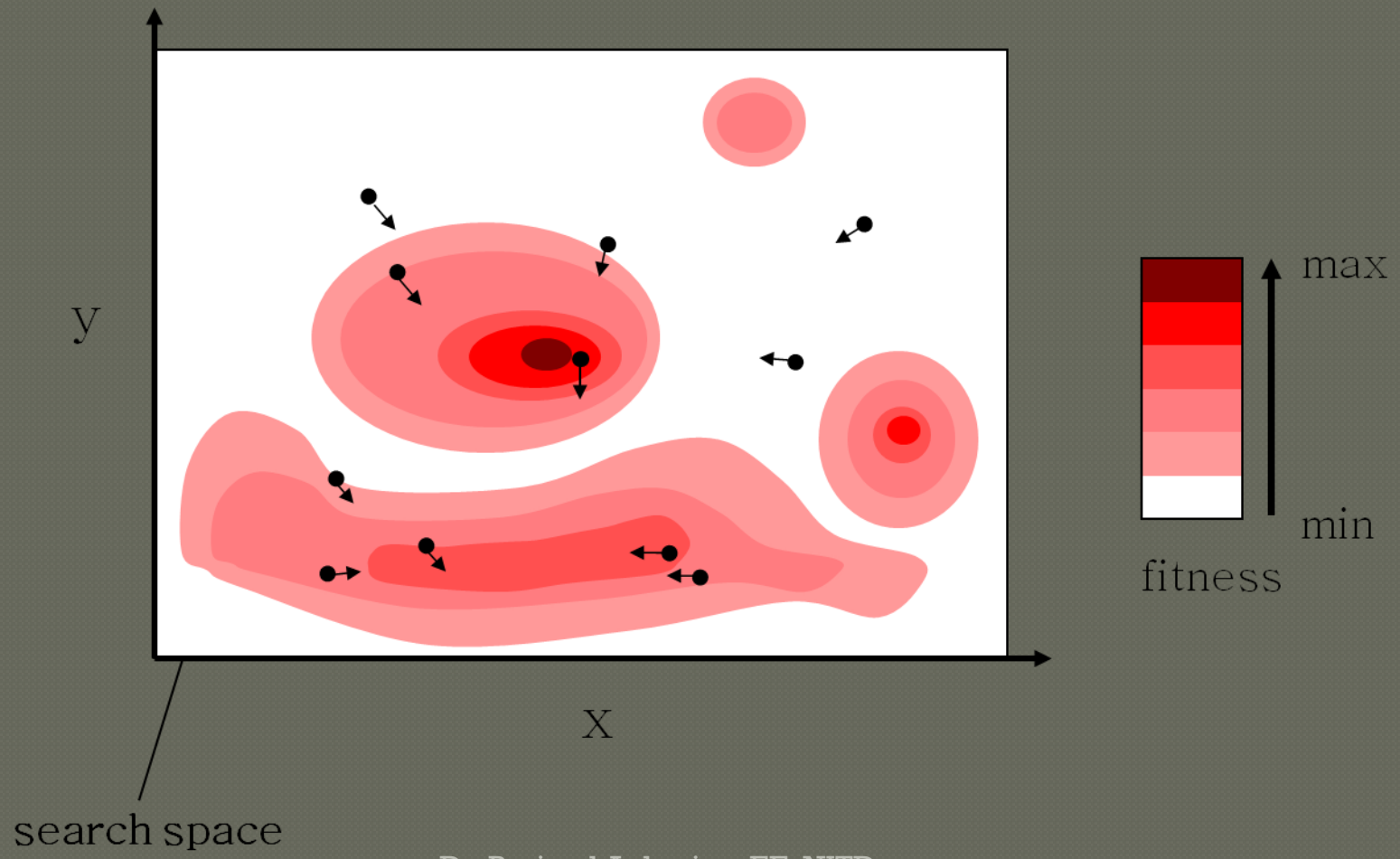
Concept



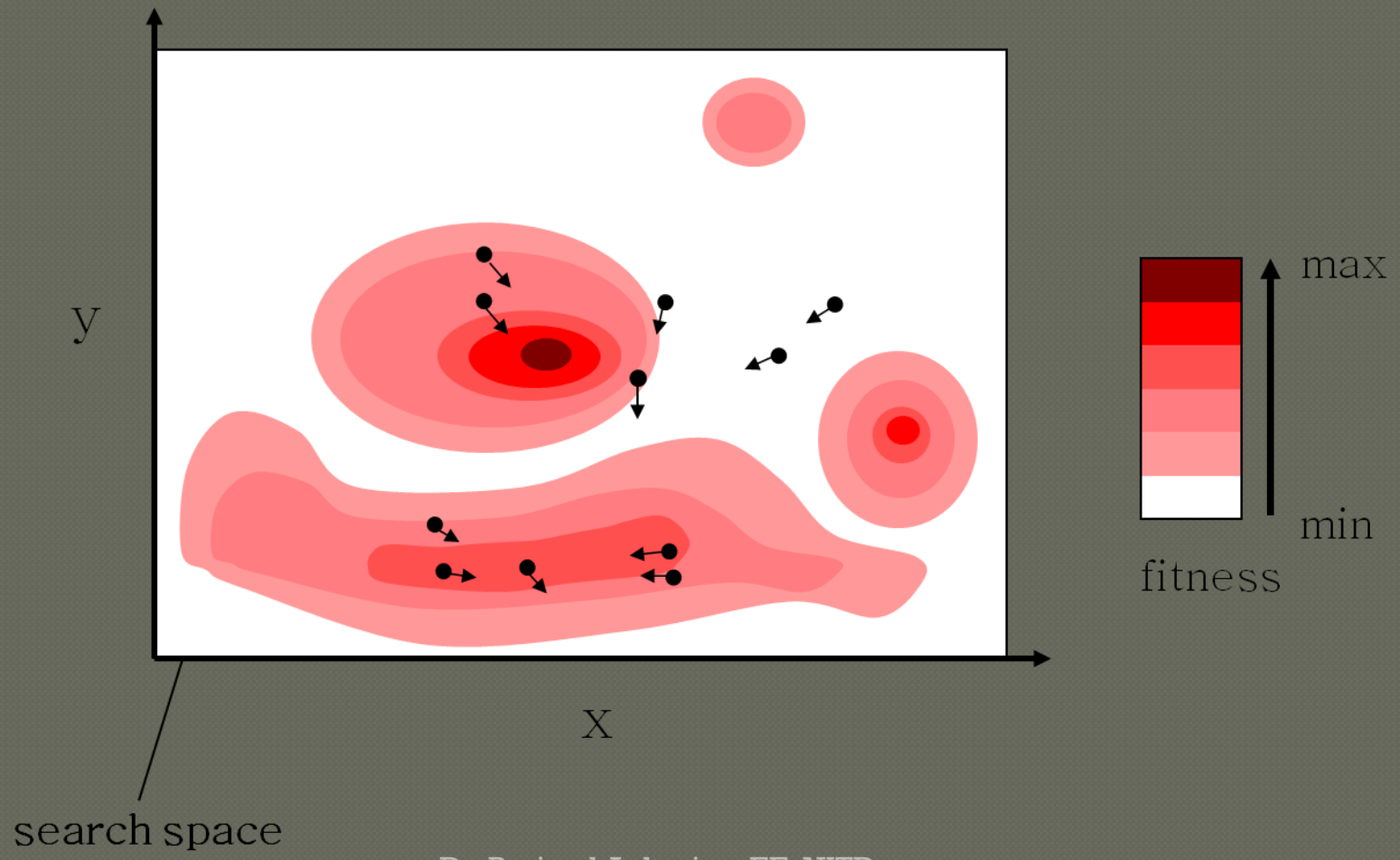
Parimal Acharjee



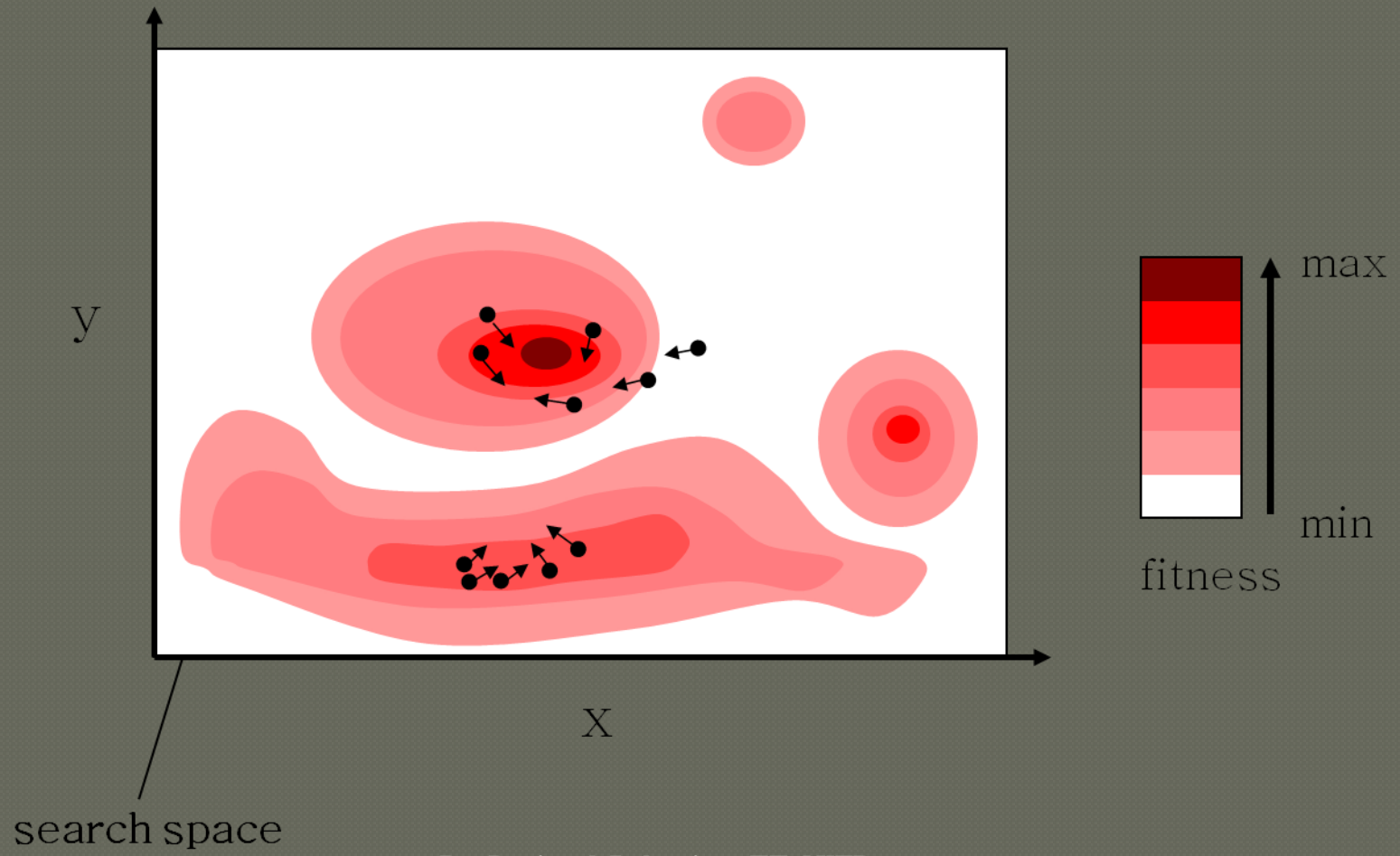
Parimal Acharjee



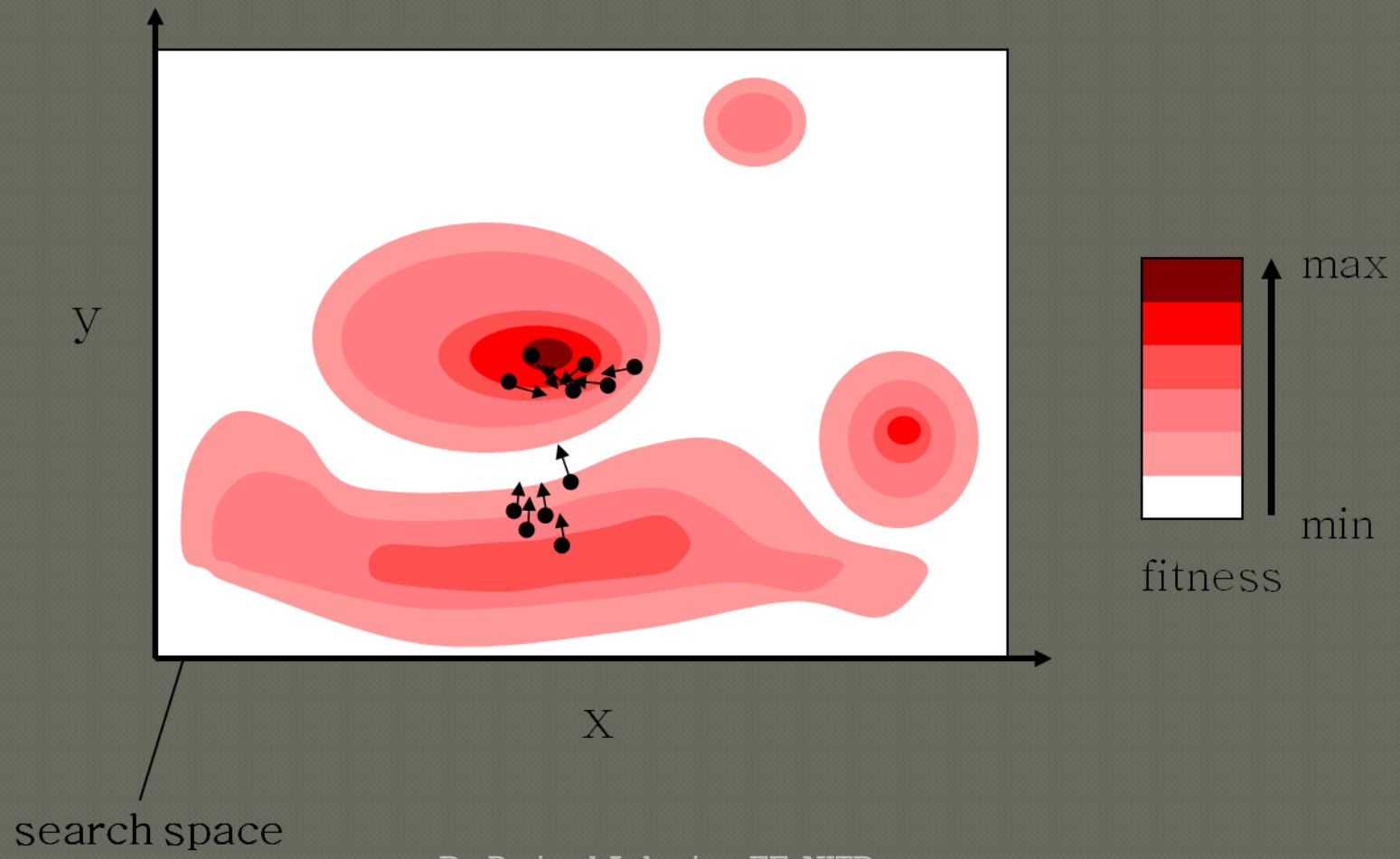
Parimal Acharjee



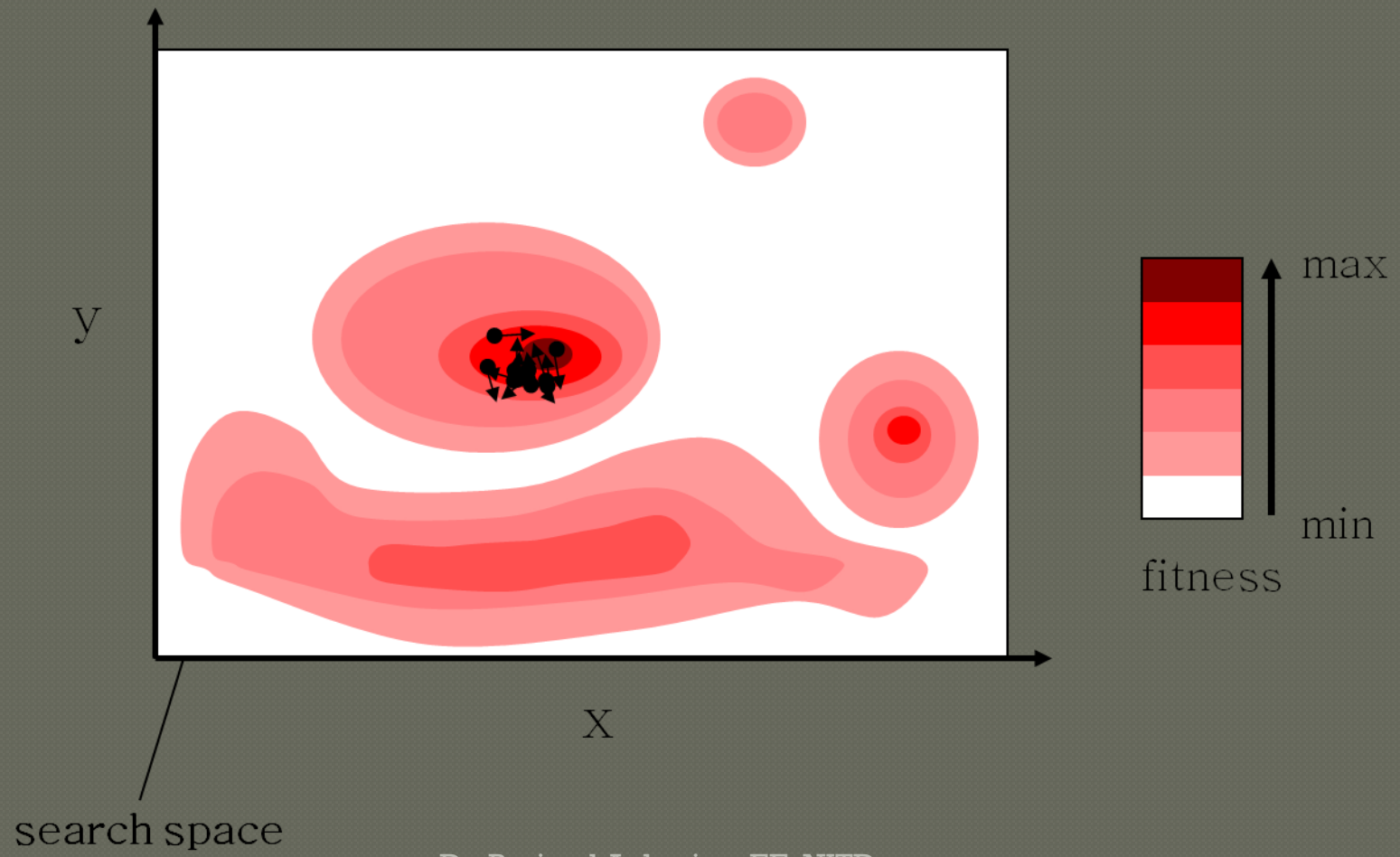
Parimal Acharjee



Parimal Acharjee



Parimal Acharjee



Parimal Acharjee

PSO INTRODUCTION

- PSO is a robust stochastic optimization technique based on the movement and intelligence of swarms.
- PSO applies the concept of social interaction to problem solving.
- It was developed in 1995 by James Kennedy (social-psychologist) and Russell Eberhart (electrical engineer).
- It uses a number of agents (particles) that constitute a swarm moving around in the search space looking for the best solution.
- Each particle is treated as a point in a N-dimensional space which adjusts its “flying” according to its own flying experience as well as the flying experience of other particles.

pbest and gbest

- Each particle keeps track of its coordinates in the solution space which are associated with the best solution (fitness) that has achieved so far by that particle. This value is called personal best , ***pbest***.
- Another best value that is tracked by the PSO is the best value obtained so far by any particle in the neighborhood of that particle. This value is called ***gbest***.
- The basic concept of PSO lies in accelerating each particle toward its pbest and the gbest locations, with a random weighted acceleration at each time step.

Basic PSO Equations

Update particles' velocities:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{\textit{inertia}} + \underbrace{c_1 \mathbf{U}_1^t (\mathbf{pb}_i^t - \mathbf{p}_i^t)}_{\textit{personal influence}} + \underbrace{c_2 \mathbf{U}_2^t (\mathbf{gb}^t - \mathbf{p}_i^t)}_{\textit{social influence}}$$

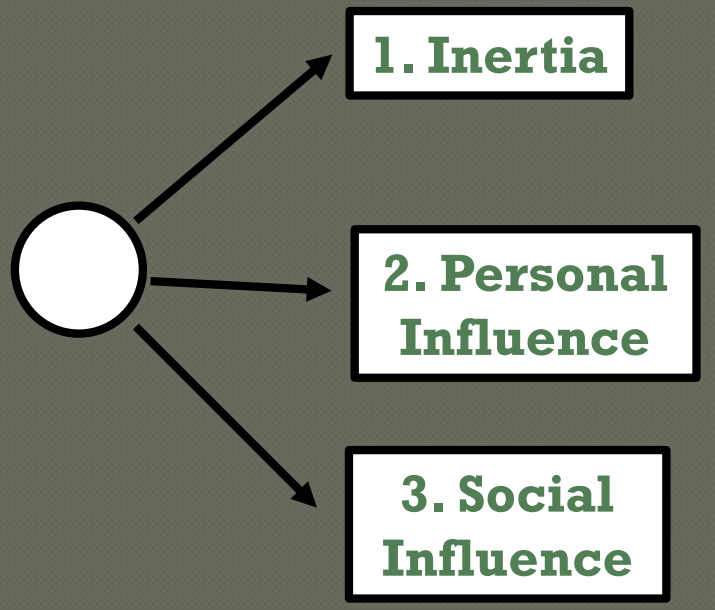
Move particles to their new positions:

$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + \mathbf{v}_i^{t+1}$$

Significance of Particle's Velocity

Particle's velocity:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{\text{inertia}} + \underbrace{c_1 U_1^t (\mathbf{pb}_i^t - \mathbf{p}_i^t)}_{\text{personal influence}} + \underbrace{c_2 U_2^t (\mathbf{gb}^t - \mathbf{p}_i^t)}_{\text{social influence}}$$



1. Inertia

- Makes the particle move in the same direction and with the same velocity

2. Personal Influence

- Improves the individual
- Makes the particle return to a previous position, better than the current
- Conservative

3. Social Influence

- Makes the particle follow the best neighbors direction

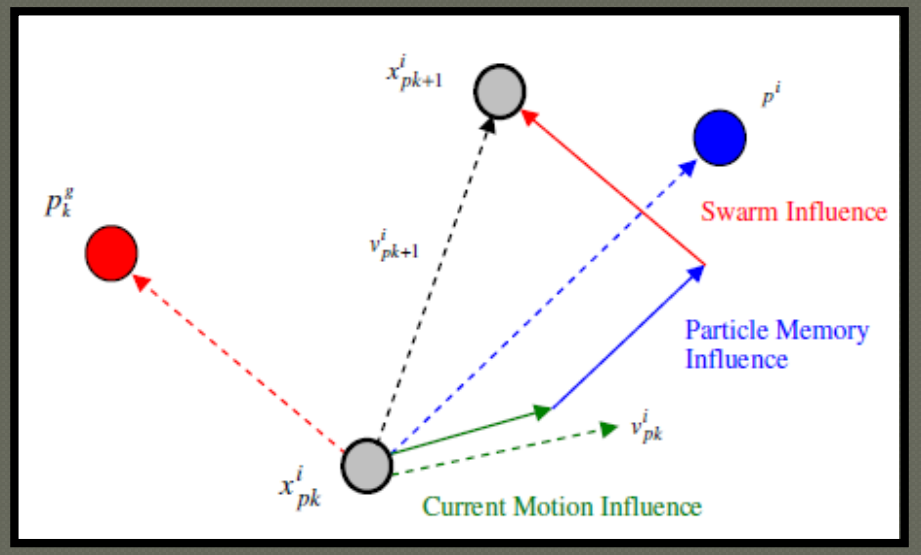
Intensification and diversification

- Intensification: explores the previous solutions, finds the best solution of a given region
- Diversification: searches new solutions, finds the regions with potentially the best solutions
- In PSO:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{\text{Diversification}} + \underbrace{c_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{p}_i^t) + c_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{p}_i^t)}_{\text{Intensification}}$$

PARTICLE UPDATATION RANDOMLY

- Each particle adjusts its travelling speed dynamically corresponding to the flying experiences of itself and its colleagues
- Each particle modifies its position according to:
  - its current position
  - its current velocity
  - the distance between its current position and *pbest*
  - the distance between its current position and *gbest*



PSO solution update in 2D

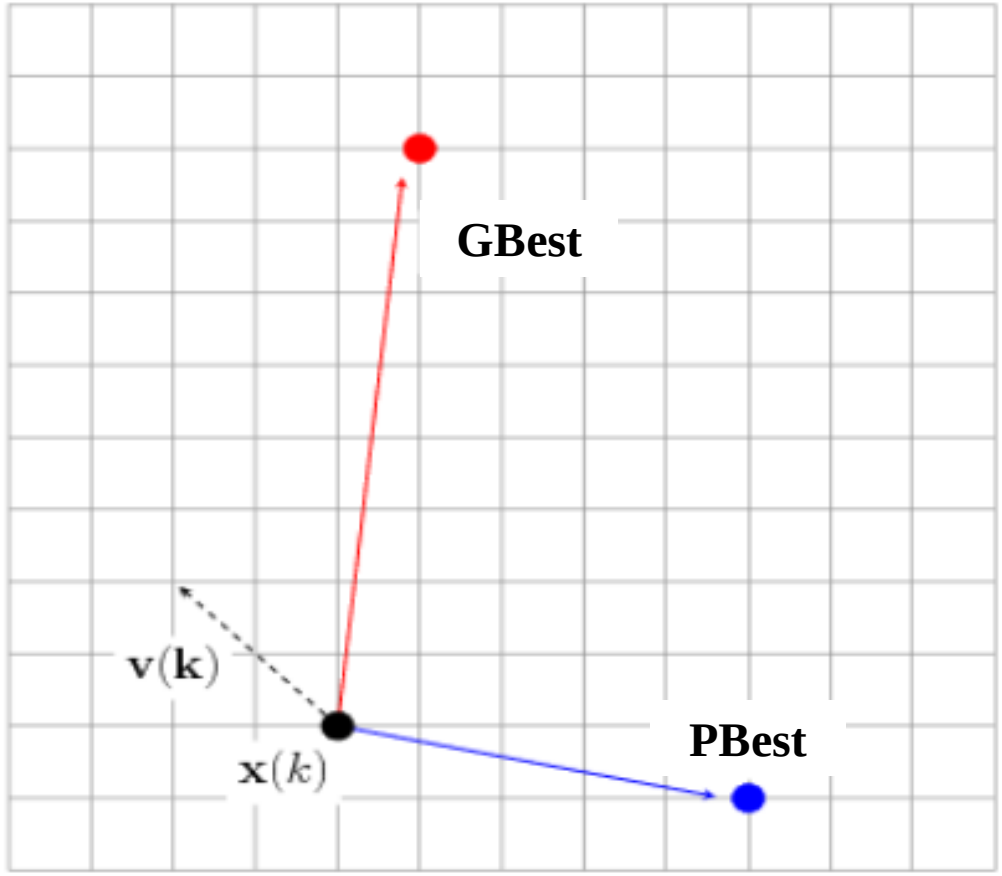
- $\mathbf{x}(k)$ - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

PSO solution update in 2D

Inertia: $\mathbf{v}(k)=(-2, 2)$

- $\mathbf{x}(k)$ - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

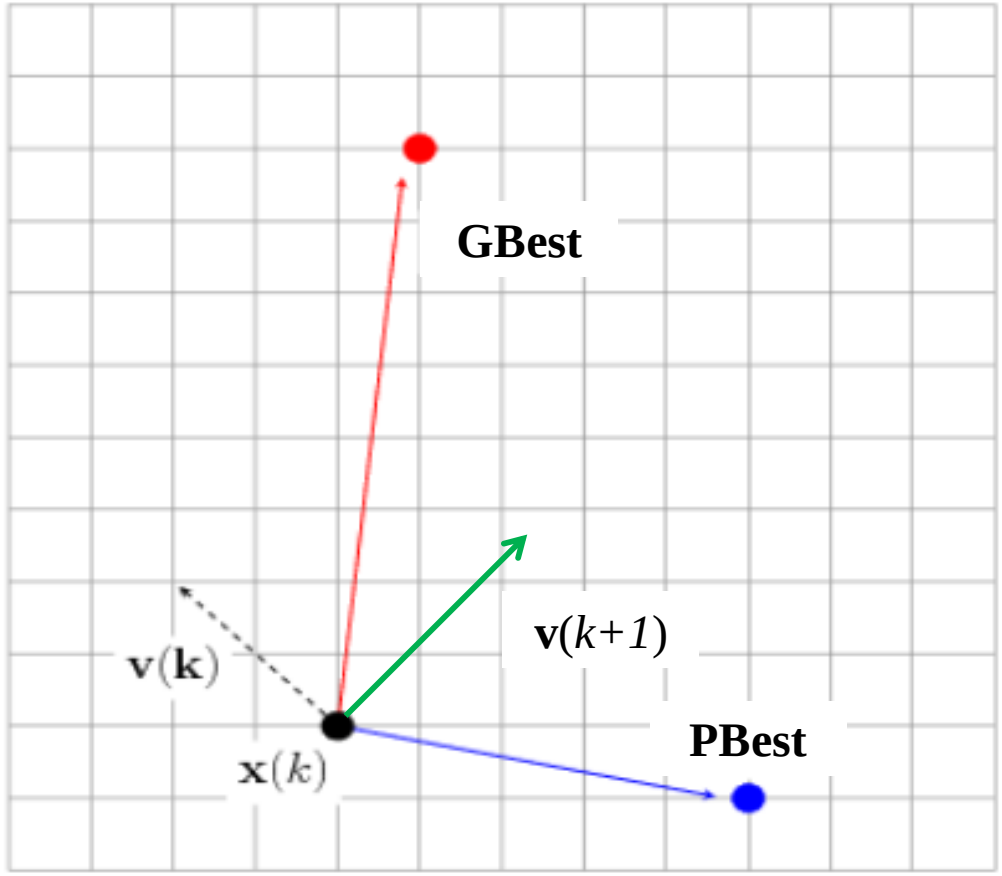
PSO solution update in 2D



- Inertia: **v**(k)=(-2,2)
- Cognitive:
  PBest-**x**(k)=(9,1)-(4,2)=(5,-1)
- Social:
  GBest-**x**(k)=(5,10)-(4,2)=(1,8)

- **x**(k) - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

PSO solution update in 2D

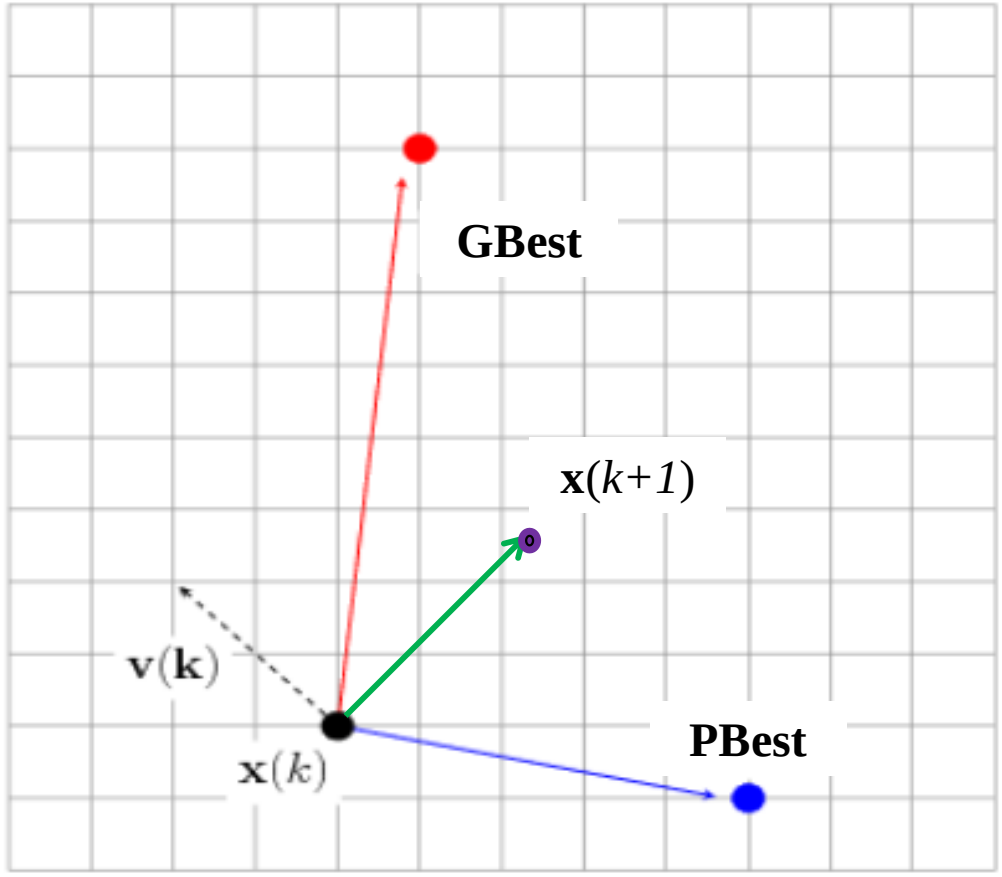


- Inertia: $\mathbf{v}(k)=(-2,2)$
- Cognitive:
  PBest-$\mathbf{x}(k)=(9,1)-(4,2)=(5,-1)$
- Social:
  GBest-$\mathbf{x}(k)=(5,10)-(4,2)=(1,8)$

$\mathbf{v}(k+1)=(-2,2)+0.8*(5,-1)$
$+0.2*(1,8) = (2.2,2.8)$

- $\mathbf{x}(k)$ - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

PSO solution update in 2D



- Inertia: $\mathbf{v}(k)=(-2,2)$
- Cognitive:
  PBest-$\mathbf{x}(k)=(9,1)-(4,2)=(5,-1)$
- Social:
  GBest-$\mathbf{x}(k)=(5,10)-(4,2)=(1,8)$
- $\mathbf{v}(k+1)=(2.2,2.8)$

$\mathbf{x}(k+1)=\mathbf{x}(k)+\mathbf{v}(k+1)=$
$(4,2)+(2.2,2.8)=(6.2,4.8)$

- $\mathbf{x}(k)$ - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

CONCEPT OF SEARCHING POINT IMPROVEMENT

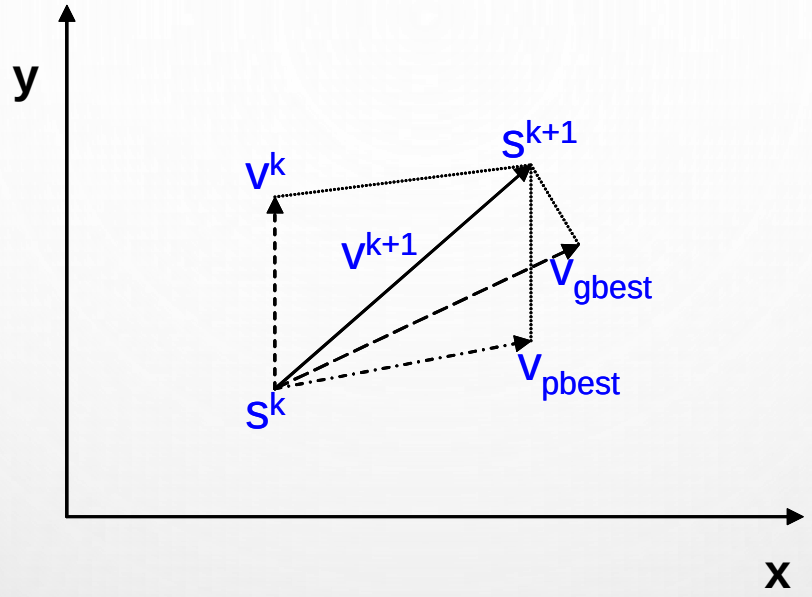


Fig.1 Concept of modification of a searching point by PSO

s^k : current searching point.
s^{k+1}: modified searching point.
v^k: current velocity.
v^{k+1}: modified velocity.
v_{pbest} : velocity based on pbest.
v_{gbest} : velocity based on gbest

UPDATED PSO EQUATION(S)

- Each particle tries to modify its position using the following information:
  - the current positions,
  - the current velocities,
  - the distance between the current position and pbest,
  - the distance between the current position and the gbest.

- The modification of the particle's position can be mathematically modeled according the following equation :

$$V_i^{k+1} = wV_i^k + c_1\, rand_1(\ldots)\ x\ (pbest_i - s_i^k) + c_2\, rand_2(\ldots)\ x\ (gbest - s_i^k) \quad (1)$$

$$s_i^{k+1} = s_i^k + V_i^{k+1} \quad (2)$$

where, v_i^k : velocity of agent i at iteration k,
w: weighting function,
c_j : Constriction factor/Learning Factor,
rand : uniformly distributed random number between 0 and 1,
s_i^k : current position of agent i at iteration k,
$pbest_i$: pbest of agent i,
gbest: gbest of the group.

ALGORITHM

1. INITIALIZE THE PARTICLES
2. CALCULATE THE FITNESS VALUE OF THE PARTICLES
3. THE BEST FITNESS VALUE IS SELECTED AND ITS CORRESPONDING PARTICLE IS TAKEN AS PBEST
4. CALCULATE PARTICLE VELOCITY
5. LIMIT THE PARTICLE VELOCITY THAT EXCEED THE MAXIMUM VELOCITY (VMAX)
6. UPDATE PARTICLE POSITION
7. CHOOSE THE PARTICLE WITH THE BEST FITNESS VALUE OF ALL THE PARTICLES AS GBEST
8. CHECK STOP CONDITION I.E. EITHER MAXIMUM ITERATION OR MINIMUM ERROR CRITERIA. IF STOP CONDITION IS SATISFIED, THEN STOP ELSE GO BACK TO STEP 2.

PARAMETER CONTROL

- NUMBER OF PARTICLES
- DIMENSION OF PARTICLES
- RANGE OF PARTICLES
- CONSTRICTION FACTORS / LEARNING FACTORS
- INERTIA WEIGHT
- MAXIMUM VELOCITY
- STOP CONDITION

PSO PROGRAM

START

INITIALIZATION

LOOP

FITNESS CALCULATION

PBEST SOLUTION SELECTION & ITS CORRESPONDING FITNESS VALUE

VELOCITY UPDATE & VELOCITY LIMIT CHECK

POSITION UPDATE & POSITION LIMIT CHECK

GBEST SOLUTION AND ITS CORRESPONDING FITNESS VALUE

CONVERGENCE CHECK

END

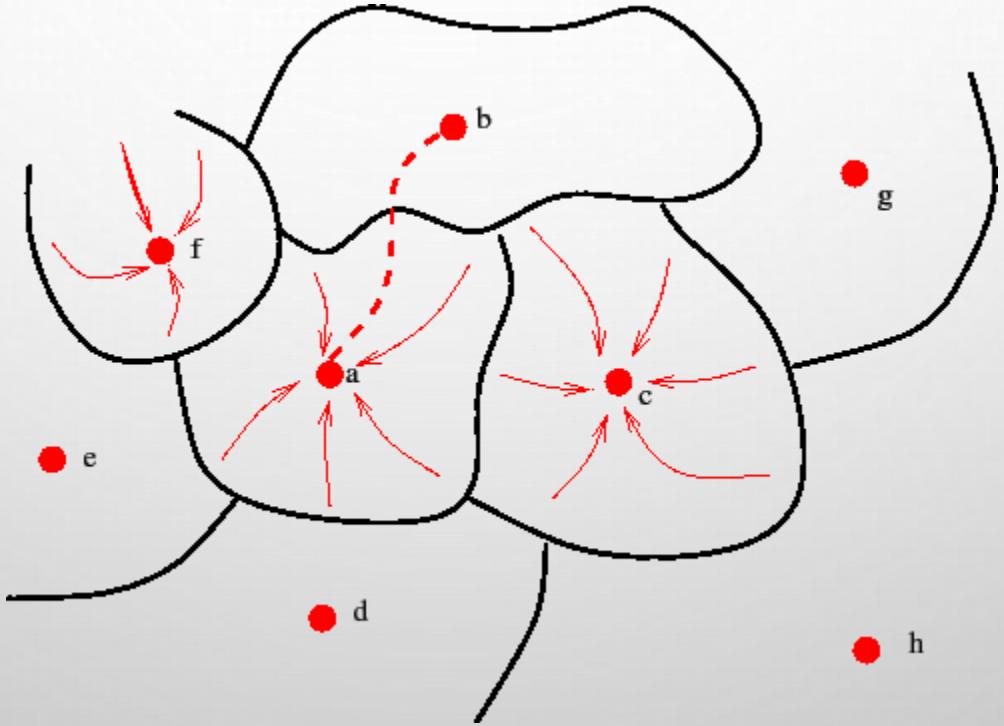
COMPARISONS BETWEEN GA AND PSO

- SIMILARITY
- DISSIMILARITY

Parimal Acharjee

LOCAL OPTIMUM

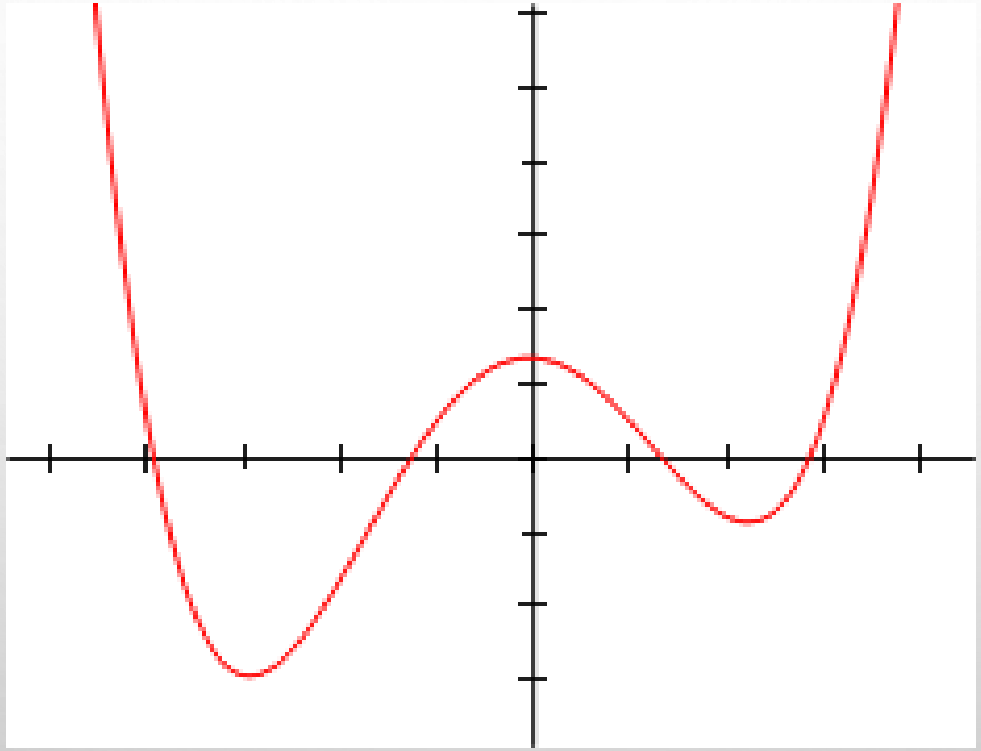
- IN APPLIED MATHEMATICS AND COMPUTER SCIENCE, A **LOCAL OPTIMUM** OF AN OPTIMIZATION PROBLEM IS A SOLUTION THAT IS OPTIMAL (EITHER MAXIMAL OR MINIMAL) WITHIN A NEIGHBORING SET OF CANDIDATE SOLUTIONS. THIS IS IN CONTRAST TO A GLOBAL OPTIMUM, WHICH IS THE OPTIMAL SOLUTION AMONG ALL POSSIBLE SOLUTIONS, NOT JUST THOSE IN A PARTICULAR NEIGHBORHOOD OF VALUES.



ATTRACTION AROUND LOCALLY OPTIMAL POINTS

GLOBAL OPTIMUM

- IN MATHEMATICS, A **GLOBAL OPTIMUM** IS A SELECTION FROM A GIVEN DOMAIN WHICH PROVIDES EITHER THE HIGHEST VALUE (THE GLOBAL MAXIMUM) OR LOWEST VALUE (THE GLOBAL MINIMUM), DEPENDING ON THE OBJECTIVE, WHEN A SPECIFIC FUNCTION IS APPLIED.



POLYNOMIAL OF DEGREE 4: THE TROUGH ON THE RIGHT IS A LOCAL MINIMUM AND THE ONE ON THE LEFT IS THE GLOBAL MINIMUM. THE PEAK IN THE CENTER IS A LOCAL MAXIMUM.

GLOBAL AND LOCAL OPTIMA

- SEARCH AREA: HILLY (MOUNTAINS) AREA
- OPTIMIZATION: PEAK SEARCHING (HIGHEST)

EXAMPLE: HIMALAYANS

- GLOBAL OPTIMA: HIGHEST PEAK (EVEREST)
- LOCAL OPTIMA: PEAK OF KANCHANJANGHA

THANK YOU

START

%PARTICLE SWARM OPTIMIZATION:
%PROBLEM: "F(X)=X1^2+X2^2+.....+X4^2“

- CLC
- CLEAR ALL

- POP=5;
- DIM=4; %DIMENSION OF THE PARTICLES/NO. OF VARIABLES
- RANGE=6; %RANGE OF THE PARTICLES
- VELOCITY_LIMIT=3; %VELOCITY LIMIT

- C1=1.49455;
- C2=1.49455; %SOCIAL CONSTANT

- MIN_INERTIA_WEIGHT=0.45;
- MAX_INERTIA_WEIGHT=0.95; %INERTIA WEIGHT RANGE

BACK

INITIALIZATION

```
%GENERTE INITIAL DATA
```

- `X=UNIFRND(-RANGE,RANGE,POP,DIM);`

```
X =

    0.5152    2.8987    0.3646    2.4965
   -4.6480    1.5373    2.0980    4.4851
   -0.3007    1.4777    2.1378    5.5079
   -4.9097    1.4279   -2.3884    3.4973
   -4.1324   -2.2955    1.2298    4.5154
```

UNIFRND

X = UNIFRND(A,B) RETURNS AN ARRAY OF RANDOM NUMBERS CHOSEN FROM THE CONTINUOUS UNIFORM DISTRIBUTION ON THE INTERVAL FROM A TO B.

X = UNIFRND(A,B,M,N)

RETURNS AN M-BY-N

POP=5; DIM=4; RANGE=6;

BACK

FITNESS CALCULATION

BACK

- FOR I=1:POP
- Y=X(I,:);
- Z=Y.^2;
- F(I)=SUM(Z);
- END

X =

0.5152	2.8987	0.3646	2.4965
-4.6480	1.5373	2.0980	4.4851
-0.3007	1.4777	2.1378	5.5079
-4.9097	1.4279	-2.3884	3.4973
-4.1324	-2.2955	1.2298	4.5154

POPULATION (POP)	FITNESS (F)
1	15.0331
2	48.4852
3	37.1815
4	44.0789
5	44.2476

PBEST SOLUTION

BACK

- [PBEST_VALUE,INDEX]=MIN(F)
- PBEST_SOLUTION=X(INDEX,:)

X =

0.5152	2.8987	0.3646	2.4965
-4.6480	1.5373	2.0980	4.4851
-0.3007	1.4777	2.1378	5.5079
-4.9097	1.4279	-2.3884	3.4973
-4.1324	-2.2955	1.2298	4.5154

SOLUTION:

- PBEST_VALUE =15.0331
- INDEX =1
- PBEST_SOLUTION =

0.5152 2.8987 0.3646 2.4965

POPULATION (POP)	FITNESS (F)
1	15.0331
2	48.4852
3	37.1815
4	44.0789
5	44.2476

VELOCITY UPDATE

BACK

```
FOR I=1:POP
    FOR K=1:DIM
        VELOCITY(I,K)=W*VELOCITY(I,K)-
C1*UNIFRND(0,1)*((X(I,K))-(PBEST_SOLUTION(K)))...
            -C2*UNIFRND(0,1)*((X(I,K))-(GBEST_SOLUTION(K)));
        IF VELOCITY(I,K)>VELOCITY_LIMIT
            VELOCITY(I,K)=VELOCITY_LIMIT;
        ELSEIF VELOCITY(I,K)<-VELOCITY_LIMIT
            VELOCITY(I,K)=-VELOCITY_LIMIT;
        ELSE
        END
    END
END
```

VELOCITY =

-2.5172	-3.0000	-0.8669	2.9144
1.1787	-2.0623	-2.9490	-2.6082
-2.1307	-3.0000	-3.0000	-2.3202
3.0000	-3.0000	3.0000	-1.5975
3.0000	3.0000	-0.5466	-2.9996

POSITION UPDATE

BACK

```
•X=X+VELOCITY;

•FOR I=1:POP
•   FOR K=1:DIM
•       X(I,K)=X(I,K)+VELOCITY(I,K);
•       IF X(I,K)>RANGE
•           X(I,K)=RANGE;
•       ELSEIF X(I,K)<-RANGE
•           X(I,K)=-RANGE;
•       ELSE
•       END
•   END
•END
```

UPDATED:

X =

-2.0020	-0.1013	-0.5023	5.4109
-3.4693	-0.5250	-0.8510	1.8768
-2.4314	-1.5223	-0.8622	3.1877
-1.9097	-1.5721	0.6116	1.8997
-1.1324	0.7045	0.6832	1.5158

X =

0.5152	2.8987	0.3646	2.4965
-4.6480	1.5373	2.0980	4.4851
-0.3007	1.4777	2.1378	5.5079
-4.9097	1.4279	-2.3884	3.4973
-4.1324	-2.2955	1.2298	4.5154

VELOCITY =

-2.5172	-3.0000	-0.8669	2.9144
1.1787	-2.0623	-2.9490	-2.6082
-2.1307	-3.0000	-3.0000	-2.3202
3.0000	-3.0000	3.0000	-1.5975
3.0000	3.0000	-0.5466	-2.9996

GBEST SOLUTION

BACK

- IF PBEST_VALUE<GBEST_VALUE
- GBEST_SOLUTION=PBEST_SOLUTION;
- GBEST_VALUE=PBEST_VALUE;
- END

PRESENT (CURRENT) ITERATION:

PBEST_VALUE =15.0331

PBEST_SOLUTION =

0.5152 2.8987 0.3646 2.4965

PREVIOUS DATA:

GBEST_VALUE = 44.2476

GBEST_SOLUTION =

-4.1324 -2.2955 1.2298 4.5154

AFTER CHECKING GBEST:

GBEST_VALUE =15.0331

GBEST_SOLUTION =

0.5152 2.8987 0.3646 2.4965

CONVERGENCE CHECK

BACK

- IF GBEST_VALUE<=.001
- RESULT=GBEST_VALUE;
- BREAK
- ELSE
- ITERATION=ITERATION+1
- END

RESULTS:

1. FINAL GBEST_VALUE
2. FINAL GBEST SOLUTION
3. CONVERGENCE RATE (NUMBER OF ITERATION FOR CONVERGENCE)
4. COMPUTATION TIME
5. CHARACTERISTIC

INERTIA WEIGHT CONTD…

$$w = w_{\max} - \frac{(w_{\max} - w_{\min}) * present\_iteration}{\max imum\_iteration} \quad (1)$$

$$w_{\max} = 0.9;\ w_{\min} = 0.4$$

$$w = 0.5 + rand()/2 \quad (2)$$

COMMENTS ON THE INERTIA WEIGHT FACTOR

A LARGE INERTIA WEIGHT (*W*) FACILITATES A GLOBAL SEARCH WHILE A SMALL INERTIA WEIGHT FACILITATES A LOCAL SEARCH.

BY LINEARLY DECREASING THE INERTIA WEIGHT FROM A RELATIVELY LARGE VALUE TO A SMALL VALUE THROUGH THE COURSE OF THE PSO RUN GIVES THE BEST PSO PERFORMANCE COMPARED WITH FIXED INERTIA WEIGHT SETTINGS.

LARGER W ---------- GREATER GLOBAL SEARCH ABILITY

SMALLER W ---------- GREATER LOCAL SEARCH ABILITY.

BACK

SIMILARITY

- FROM THE PROCEDURE, WE CAN LEARN THAT PSO SHARES MANY COMMON POINTS WITH GA.
- BOTH ALGORITHMS START WITH A GROUP OF A RANDOMLY GENERATED POPULATION;
- BOTH HAVE FITNESS VALUES TO EVALUATE THE POPULATION.
- BOTH UPDATE THE POPULATION AND SEARCH FOR THE OPTIMUM WITH RANDOM TECHNIQUES.
- BOTH SYSTEMS DO NOT GUARANTEE SUCCESS.

BACK

DISSIMILARITY

- IN PSO THE INFORMATION SHARING MECHANISM IS SIGNIFICANTLY DIFFERENT FROM GA.
- IN GA, THE WHOLE POPULATION MOVES LIKE A ONE GROUP TOWARDS AN OPTIMAL AREA AS CHROMOSOMES SHARE INFORMATION WITH EACH OTHER.
- PSO IS ONE-WAY INFORMATION SHARING MECHANISM. HERE GBEST/PBEST GIVES THE INFORMATION TO THE OTHERS.
- CHANGE IN GENETIC POPULATIONS RESULTS IN DESTRUCTION OF PREVIOUS KNOWLEDGE OF THE PROBLEM.
- IN PSO, INDIVIDUALS WHO FLY PAST OPTIMA ARE TUGGED TO RETURN TOWARD THEM, GOOD SOLUTIONS ARE RETAINED BY ALL THE PARTICLES. PSO ONLY LOOKS FOR THE BEST SOLUTION. ALL THE PARTICLES HAVE A TENDENCY TO CONVERGE TO THE BEST SOLUTION QUICKLY IN MOST CASES.

BACK