NEURAL NETWORK UNIT-3

What is Backpropagation -

Back-propagation is the essence of neural net training. It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e, iteration)

Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization.

⇒ Backpropagation is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks. this method helps to calculate the gradient of a loss function with respects to all the weights in the network.

→ It is an algorithm for supervised learning of ANN using gradient descent

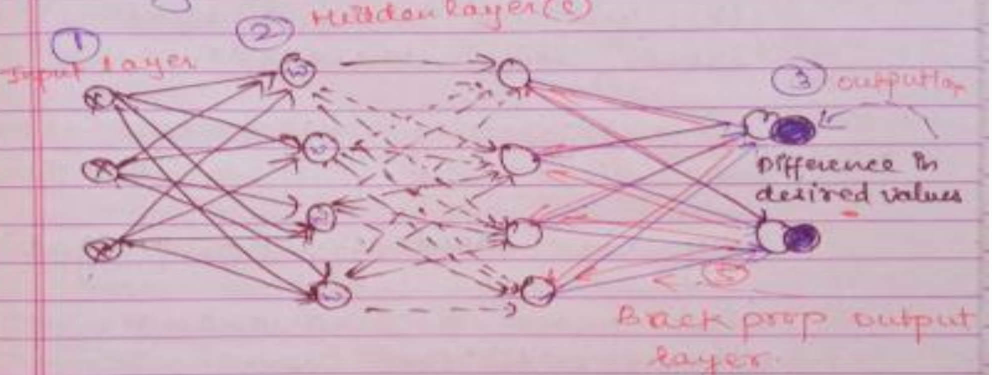
Page No.:
Date: / /

(2)

How Backpropagation works:

Simple algorithm-

- consider the following propagation diagram



1) Inputs X, arrive through the preconnected path

2) Input is modeled using real weights W. the weights are usually randomly selected.

3) calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.

4. Calculate the error in the outputs

$\boxed{Error_B = Actual\ Output - Desired\ output}$

5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

Keep repeating the process until the desired output is achieved.

Why We Need Backpropagation?

- Most prominent advantages of Backpropagation are.

- Backpropagation is fast, simple and easy to program
- It has no parameters to tune tune apart from the numbers of input
- It is a flexible method as it does not require prior knowledge about the network.

- It is a standard method that generally works well
- It does not need any special mention of the features of the function to be learned.

What is a feed forward Network?

— A feedforward neural network is an artificial neural network where the nodes never form a cycle. this kind of neural network has an input layer, hidden layer and an output layer. It is the first and simplest type of artificial neural network.

Types of Backpropation Network-

two types of Backpropagation Networks are:-

- Static Back-propagation
- Recurrent Backpropagation.

Static back-propagation-

It is one kind of back propagation network which produces a mapping of a static inputs for static output. It is useful to solve static classification issues like optical character recognition.

Recurrent Backpropagation-

Recurrent backpropagation is fed forward until a fixed values is achieved. After that, the error is computed and propagated backward

The main difference between both of these method is: that the mapping is rapid in static back-propagation while it is nonstatic in recurrent backpropagation

(5)

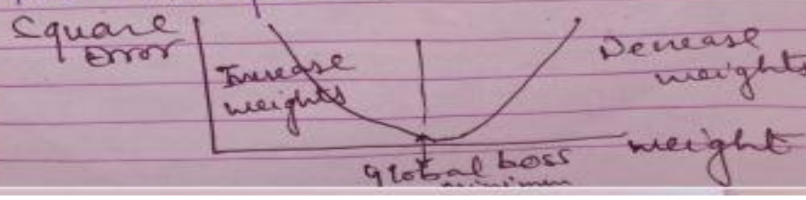
Page No.:
Date: / /

Disadvantages of using Backpropagation :-

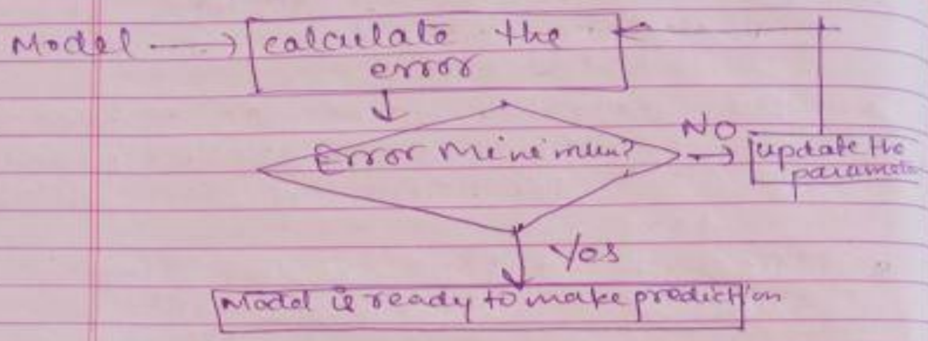
- The actual performance of back-propagation on a specific problem is dependent on the input data.
- Backpropagation can be quite sensitive to noisy data.
- You need to use the matrix based approach for background propagation instead of mini-batch.

Gradient descent

- Update the weights using gradient descent.
- Gradient descent is used for finding the minimum of a fn.
- In our case we want to minimize the error function.



Back propagation



the goal of training a neural network model is most challenging because it requires solving two hard problems at once.

- Learning – Learning the training in order to best minimize the loss.
- Generalizing – Generalizing the model performance in order to make predictions on unseen examples.

Some important terms —

when fitting a neural network model these terms can be defined as:

- Bias :- A measure of how the network output averaged across all datasets differ from the desired function.

- Variance :- A measure of how much the network output varies across data sets.

Heuristic for making BP-algorithm perform better. —

The heuristic method is widely used to improve the convergence rate of training the BP algorithm, and includes two parameters, namely the training rate and the "momentum" coefficient.

— The heuristic method is very important to increase the training algorithm.

1) - A MLP trained with the back propagation algorithm may, in general learn faster (in terms of the number of training iterations required)

when the asymmetric sigmoidal activation function are used in neuron model, than when it is non-symmetric.

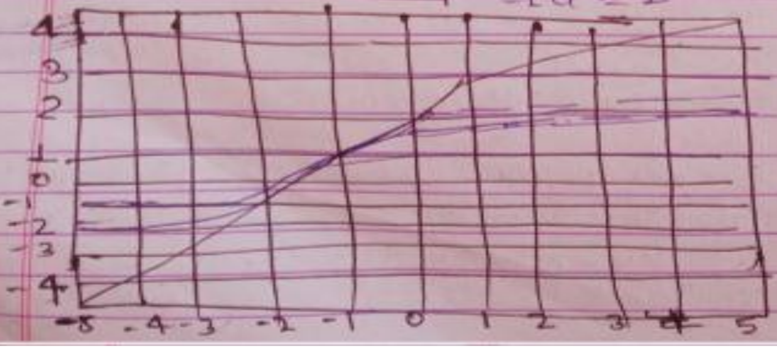
Assymmetric function: $\varphi(-V) = -\varphi(V)$

2) It is important that the desired value are chosen within the range of the sigmoid activation functions

otherwise, the back-propagation algorithm tends to drive the free parameters of the network to infinity, and thereby slow down the learning process by orders of magnitude

$$\boxed{\frac{1}{a} = a + \underset{\text{offset}}{e}}$$

$a = 0.2$
$-2a = 0.5$
$-1a = 1$



Page No.:
Date: / /

3) The initialization of the synaptic weights and threshhold levels of the network should be uniformaly distributed inside a small range.

The reason for making the range small is to reduce the likelihood of the neurons in the network saturating and producing small error gradients.

However, the range should not be made too small, as it can cause the error gradients to very small and the learning therefore to be initially very slow.

4) All neurons in the multilayer Perceptron should desirably learn at the same rate.

5) For on-line operation, pattern-by-pattern updating rather than batch updating should be used for weight adjustment.

- pattern-by-pattern updating tends to be orders of magnitude faster than batch updating.

6) The order in which the training examples are presented to the network should be randomized (shuffled) from ~~#~~ one epoch to the next.

This form of randomization is critical for improving the speed of convergence.

Approximation properties of RBF networks —

The Radial basis function (RBF) network has its foundation in the conventional approximation theory.
- It has the capability of universal approximation.

The RBF network is a popular alternative to the well-known multilayer perceptron (MLP). Since it has a simpler structure and a much faster training process.

Introduction -

the multilayer perceptron (MLP) trained with backpropagation (BP) rule is one of the most important neural network models. Due to its universal function approximation capability, the MLP is widely used in system identification prediction, regression, classification, control, feature extraction, and associati memory.

- ~~the RBF network has equivalent capabilities as the M~~

- The RBF network model was proposed by Broomhead and Lowe in 1988. It has its foundation in the conventional approximation techniques, such as generalization splines and regularization techniques.

- the RBF network has equivalent capabilities as the MLP model, but with a much faster training speed, and thus it has become a good alternative to the MLP.

- The RBF network has its origin in performing exact interpolation of a set of data points in a multidimensional space.

- **Network Architecture -**

The RBF network is a three-layer ($J_1 - J_2 - J_3$) feedforward neural network, as shown in figure.

1) Each node in the hidden layer use a radial basis function (RBF), denoted $\phi(r)$, as its nonlinear activation function.
The hidden layer performs a nonlinear transform of the input and the output layer is a linear combiner mapping the nonlinearity into a new space. Usually the same RBF is applied on all nodes: that is, the RBF nodes have the nonlinearity

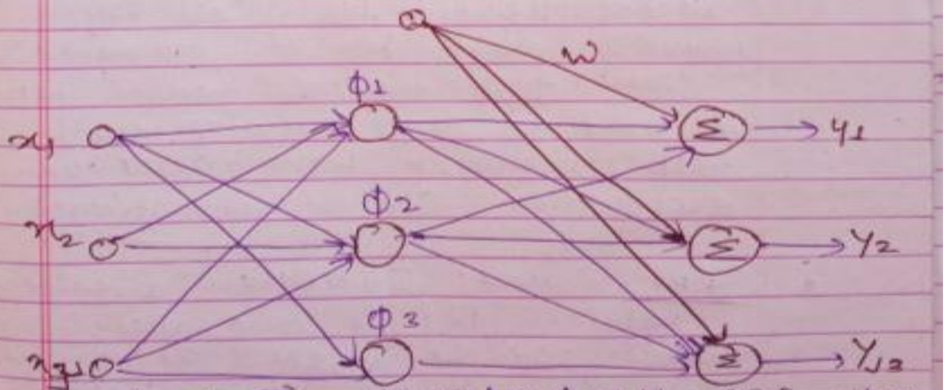
$$\phi_i(\vec{x}) = \phi(\vec{x} - \vec{c_i}),$$
$$i = 1, \dots, J_2,$$

where $\vec{c_i}$ is the prototype or center of the ith node an $\phi(\vec{x})$ is an RBF.

- The RBF network achieves a global optimal sollution to the adjustable weights in the minimum mean square error (MSE) sense by using the linear optimization method.

Figure-

Architecture of the RBF network. The input hidden, and output layers have J_1, J_2 and J_3 neurons respectively. $\phi_0(\vec{x}) = 1$ corresponds to the bias in the output layer, while $\phi_i(\vec{x})$'s denote the nonlinearity at the hidden nodes.



For input $\vec{x}$, the output of the RBF network is given by

$$y_i \boxtimes \vec{x} \boxtimes \sum_{k=1}^{J_2} w_{ki} \phi \boxtimes \| \vec{x} \to \vec{c}_k \| \boxtimes, \quad i = 1, \ldots J_3.$$

where $y_i(\vec{x})$ is the ith output, w_{ki} is the connection weight from the k th hidden unit to the ith output unit and $\|\cdot\|$ denotes the Euclidean norm

RPF Network Comparision with Multi-layer Perceptron (MLP) —

- MLP - uses dot products (between inputs and weights) and sigmoidal activation functions (or other monotonic functions such as Relu and training is usually done through backpropagation for all layers (which can be as many as you want) This type of neural network is used in deep learning with the help of many techniques (such as dropout or batch normalization).

- RBF :- uses Euclidean distances (between inputs and weights, which can be viewed as centers) and (usually) Gaussian activation functions (which could be multivariate), which makes ~~neurons~~ neurons more locally sensitive. Thus, RBF neurons have maximum activation when the center/ weights are equal to the inputs (look at the image below).

Due to this property, RBF neural networks are good for novelty detection (if each neurons is centered on a training example, inputs far away form all neurons constitute novel patterns) but not so good at extrapolation.

Also, RBFs may use backpropagation for learning, or hybrid approaches with unsupervised learning in the hidden layer.
Finally, RBFs make it easier to grow new neurons during training.

- RBF and MLP belong to a class of nenoorks called feed-forward networks.
- Hidden layer of RBF is different from MLP.

Page No.:
Date: / /

Ques- An accelerated learning algorithm-

An accelerated algorithm (ABP-adaptive back propagation) is proposed for the supervised training of multilayer perceptron networks. The learning algorithm is inspired from the principle of "forced dynamics" for the total error functional.
The algorithm updates the weights in the direction of steepest descent but with a learning rate a specific function of the error and of the error gradient norm.
This specific form of this function is chosen such as to accelerate convergence furthermore, ABP introduce no additional "tuning" parameters found in variants of the backpropagation algorithm. Simulation results indicate a superior convergence speed for analog problems only, as compared to other competing methods as well as reduced sensitivity to algorithm step size parameter variations.

Page No.: Date: / /

Quickprop - algorithm

eqn (1) - Error derivative at previous epoch

$$\frac{\delta E(n-1)}{\delta w_{ij}(n-1)} \quad \text{--- (1)}$$

eqn (2) Error derivative at this this epoch

$$\frac{\delta E(n)}{\delta W_{ij}(n)}$$

The Quickprop algorithm is loosely based on Newton's method It is quicker than standard backpropagation because it uses an approximation to the error curve, and second order derivative information which allow a quicker evaluation

the training is similar to backprop except for a copy of eqn (1) the error derivative at a previous epoch. this, and the this current error derivative (eq 2) are used to minimize an approximation to this error curve.

(18)

Quickprop update Rule —

$$\Delta w_{ij}(n) = \frac{\nabla E(w_n)}{\nabla E(w_{n-1}) - \nabla E(w_n)} \Delta w_{ij}(n-1)$$

⇒ <u>Rprop</u> – (Resilent Back propagation)

It's more complex than back propagation but Rprop has advantages in training speed and efficiency

Resilent back propagation (RProp), an algorithm that can be used to train a neural network. is similar to the more common back-propagation But it has two main advantages over back propagation.

1) training with back propagation is often faster than training with back propagation.

2) RProp doesn't require you to specify any free parameter values as opposed to back propagation which needs values for the

learning rate (and usually an optional momentum term)

Disadvantages

The main disadvantages of RProp is that its a more complex algorithm to implement than back propagation.

The Rprop algorithm makes two significant changes to the back-propagation.

1) Rprop doesn't use the magnitude of the gradient to determine a weight delta instead, it uses only the sign of gradient.

2) Instead of using a single learning rate for all weights and biases, Rprop maintains seprate weight deltas for each weight and bias, and adapts these deltas during training

Page No.:
Date: / /

Recursive least squares (RLS) learning algorithm -

The recursive least squares (RLS) learning algorithm for multilayer feedforward neural networks uses a sigmoid nonlinearity at node outputs. It is shown that by using a piecewise linear function at node outputs, the algorithm becomes faster. The modified algorithm improves computational efficiency and by preserving matrix symmetry it is possible to avoid explosive divergence, which is normally seen in the conventional RLS algorithm due to the finite precision effects. Also the use of the piecewise linear function avoids the approximation, which is otherwise necessary in the derivation of the conventional algorithm with sigmoid nonlinearity.

UNIT-IV

(1)

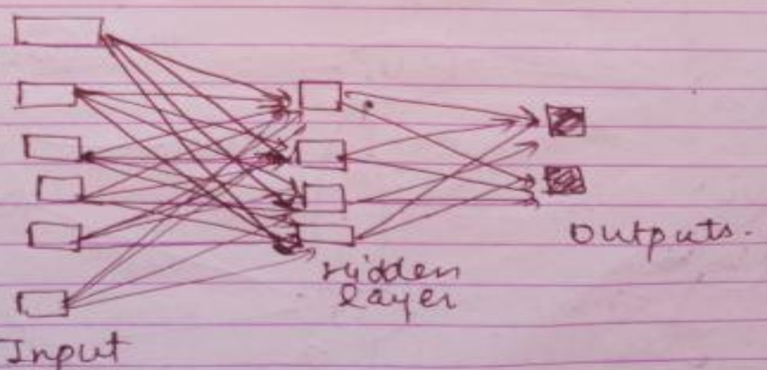
Page No.:
Date: / /

- **Reccurrent network and temporal feed-forward network**

feed forward

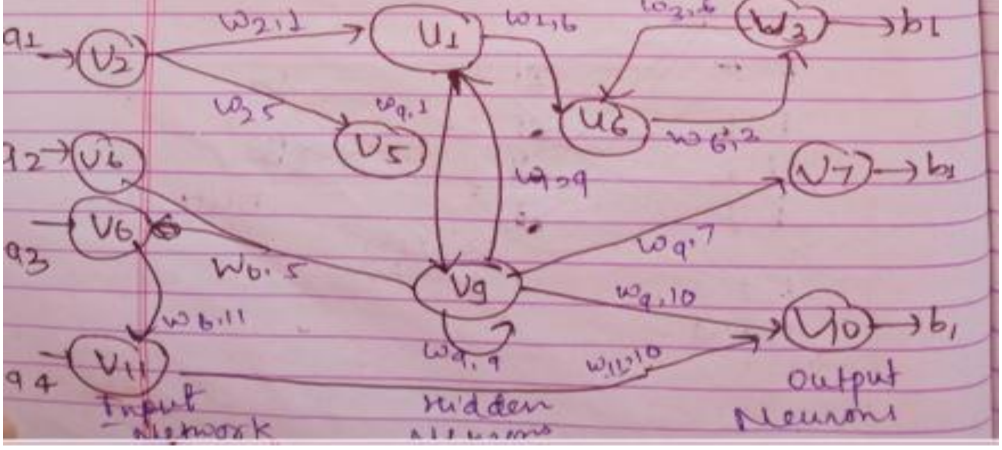
feed-forward ANNs allow signals to travel one way only: from input to output. there are no feedback (loops); i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straightforward networks that associate inputs with outputs. They are extensively used in pattern recognition.

This type of organization is also reffered to as bottom-up or top-down.



<u>feedback</u> - (or recurrent or interactive) feedback networks can have signals traveling in both directions by introducing loops in the networks are powerful and can get extremely complicated. computations derived from earlier input are fed back into the network which gives them a kind of memory.

- Feedback networks are <u>dynamic</u> their 'state' is changing continuously until they reach an equilibrium point. the remain at the equilibrium point until the input changes and a new equilibrium needs to be found.



feedforward neural networks are ideally suitable for modeling relationship b/w as set of predictor or input variables and one or more response or output variables.

Self-Organizing Map (SOM)-

It is a very useful technique for clustering analysis and exploring data.

A self-organizing map (SOM) is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction

→ Self-organising maps differ from other artificial neural networks as they apply competitive learning as opposed to error-correction learning (such as backpropagation with gradient descent), and in the sense that they use a neighborhoop function to preserve the topological properties of the input space.

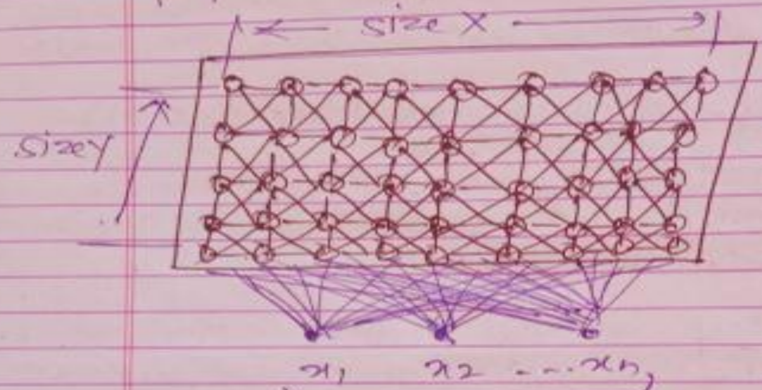


fig - Dimencionality reduction in SOM

SOM was introduced by Finish professoror Teuvo Kohonen in the 1980s is sometimes called a <u>Kohonenmap</u>.

The Algorithm:-

1. Each node's weights are initialized
2. A vector is chosen at random from the set of training data.

3. Every node is examined to calculate which one's weights are most like the input vector. The winning node is commonly known as the Best Matching unit (BMU).

4. Then the neighbourhood of the BMU is calculated. The amount of neighbors decreases over time.

5. The winning weight is rewarded with becoming more like the sample vector. The neighbours also come more like the sample vector.

The closer a node is to the BMU, the more its weights get altered and the farther away the neighbors is from the BMU, the less is learnt.

Best matching Unit is a technique which calculates the distance from each weight to the sample vector, by running through all weight vectors. The weight with the shortest distance is the winner.
There are numerous ways to determine the distance, however the most commonly used method is the Euclidean Distance.

Implementation -

Comming to the implementation part, there are various Phython libraries (minisom, sompy) out there which you could directly use to implement SOM.

II Cons:-

1- It does not build a generative model for the data, i.e. the model does not understand how data is created.

2. It does not behave so gently when using categorical data, even worse for mixed types data.

3. The time for preparing model is slow, hard to train against slowly evolving data.

Feature Map — Convolution Neural Networks (CNN) are special type of feed-forward Artificial Neural Networks that are generally used for image detection tasks.

In a convolutional neural network units within a hidden layer are segmented into "feature maps" where the units within a feature map share the weight matrix, or in simple terms look for the same feature.
The hidden units within a feature map are unique in that they are connected to different units in the lower layer so for the

first hidden layer, units within a feature map will be connected to different regions of the input

images.
so in summary, a hidden layer is segmented into feature maps where each unit in a feature map looks for the same feature but at different positions of the input image.

- Principal component analysis

Principal component analysis is dimensionality reduction method that is usually used to reduce large number of input variable to a small number of variables that still contains most of information as a large dataset

Steps of Principal component Analysis -

Step 1 - Look the features and for the observations. Let we have m feature and n observation.

Page No.:
Date: / /

Step 2 - Standardized the data.

Step 3 - calculate eigenvalue and eighen vector of covariance matrix

Step 4 - calculate eigen value and variance explained by principal component

Step 5 - Derive the new data through the selected principal components.

(new = eigen vector × Data)

Use of principal component analysis:-

- To Reduce the feature.
- to handle missing values

Mathematical proof of principal component analysis.

Lets assume we have a predictors. which are X and Y. we have 2 observation for each

X	Y
1	2
2	1

First we will make this data into standardized form.

Mean of $X = 1.5$
Mean of $Y = 1.5$

standardized $X =$

$X - mean(X)$	$Y - mean(Y)$
-0.5	0.5
0.5	-0.5

$$\text{cov}(X,Y) = \frac{\sum_{i=1}^{n}(X - mean(X)) * (Y - mean(Y))}{n-1}$$

covariance matrix $X = A = \begin{bmatrix} \text{cov}(X,X) & \text{cov}(X,Y) \\ \text{cov}(Y,X) & \text{cov}(Y,Y) \end{bmatrix}$

$$X = \begin{bmatrix} (0.25+0.25)/2 & -(0.25+0.25)/2 \\ -(0.25+0.25)/2 & (0.25+0.25)/2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.25 & -0.25 \\ -0.25 & 0.25 \end{bmatrix}$$

lets find the eigen value and eigen vector-

$(A - \lambda I) z = 0$

As we know that z is eigen vectors so it can't be zero so

$$A - \lambda I = 0 \iff \begin{bmatrix} 0.25 & -0.25 \\ -0.25 & 0.25 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 0.25 - \lambda & -0.25 \\ -0.25 & 0.25 - \lambda \end{bmatrix}$$

$$= (0.25 - \lambda) * (0.25 - \lambda) - 0.25 * 0.25$$

$$= 0.0625 - 0.50\lambda + \lambda^2 - 0.0625$$

$$= \lambda^2 - 0.50\lambda$$

$b = -0.50$

$a = 1$

$c = 0$

$$\text{roots} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{0.50 \pm \sqrt{0.50 * 0.50 - 0}}{2}$$

$$= \frac{0.50 \pm 0.50}{2}$$

$\text{root} = 1 = \lambda_1 = 0$

$\text{root } 2 = \lambda_2 = 0.50$

eigen vector corresponding to eigen value λ_1

$$(A - 0 * I) * z = 0 \iff \begin{bmatrix} 0.25 & -0.25 \\ -0.25 & 0.25 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

$0.25z_1 - 0.25z_2 = 0 \iff$

$z_1 = z_2$

Page No.
Date : / /

Independent component analysis —

ICA is a machine learning technique to seprate independent sources from a mixed signal. Unlike principal component analysis which focuses on maximizing the variance of the data points. The independent component analysis which focuses on maximizing the variance of the data points, the independent component analysis focuses on independence, i.e., independent components.

- **Problems**:- to extracts independent sources signals from a mixed signal composed of the signals from those sources.

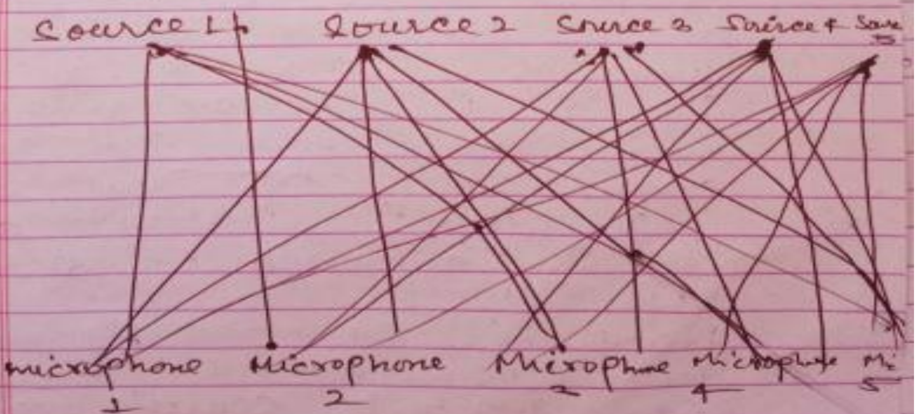
Given:- Mixed signal from five different sources independent sources.

Aim - to decompose the mixed signal into independent sources.

- source 1
- source 2
- Source 3
- source 4
- source 5

Solution: Independent Component Analysis (ICA)

Consider cocktail party problem or Blind source separation problem to understand the problem which is solved by independent component analysis.



Here, there is a party going into a room full of people. There is 'n' number of speakers in that room and they are speaking simultaneously at the party. In the same room, there are also 'n' number of microphones placed at different distances from the speakers is equal to the number of speakers is equal to the number must of microphones in the room.

Now, using these 'n' microphones recordings, we want to seprate all the room. given each microphone recorded the voice signals in the room give each microphone recorded the voice signals coming from each speaker of different intensity due to the difference in distance b/w them.

Decomposing the mixed signal of each microphone's recording into indepent source's speech signal of each microphone's recording into independent sources speech signal can be done by using the machine learning techniques independent componant analysis

$[x_1, x_2, \ldots x_n] \Rightarrow [y_1, y_2, \ldots y_n]$
where $x_1, x_2 \ldots x_n$ are the original signals present in the mixed signal and $y_1, y_2 \ldots y_n$ are the new features and are independent component analysis. which are independent of each other.

Difference between PCA and ICA —

Principal componant analysis	Independent component analysis
It reduces the dimensions to avoid the problem of overfitting	It decomposes the mixed signal into its independent sources signals.
It deals with the principal components	It deals with the Independent components

16

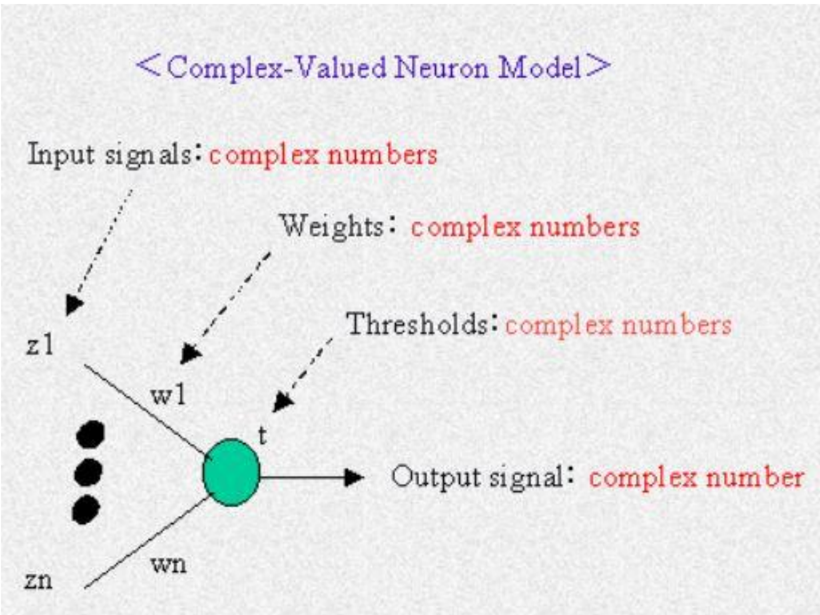
Page No.:
Date: / /

• It focuses on maximizing the variance	• It doesn't focus on the issue of variance among the data points
• It focuses on the mutual orthogonality property of the principal components.	• It does't focus on the mutual orthogonality of the components

UNIT V

Complex valued NN and complex valued BP

The complex-valued Neural Network is an extension of a (usual) real-valued neural network, whose input and output signals and parameters such as weights and thresholds are all complex numbers (the activation function is inevitably a complex-valued function).



Neural Networks have been applied to various fields such as communication systems, image processing and speech recognition, in which complex numbers are often used through the Fourier Transformation. This indicates that complex-valued neural networks are useful. In addition, in the human brain, an action potential may have different pulse patterns, and the distance between pulses may be different. This suggests that introducing complex numbers representing phase and amplitude into neural networks is appropriate. In these years the complex-valued neural networks expand the application fields in image processing, computer vision, optoelectronic imaging, and communication and so on. The potentially wide applicability yields new aspects of theories required for novel or more effective functions and mechanisms.

The learning speed of the complex-valued back-propagation learning algorithm (called Complex-BP) for multi-layered complex-valued neural networks is 2 or 3 times faster than that of the real-valued one (called Real-BP). In addition, the required number of parameters such as the weights and the thresholds is only about the half of the real-valued case.

What is Activation Function?

It's just a thing function that you use to get the output of node. It is also known as Transfer Function.

Why we use Activation functions with Neural Networks?

It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).

The Activation Functions can be basically divided into 2 types

1. Linear Activation Function
2. Non-linear Activation Functions

Linear or Identity Activation Function

As you can see the function is a line or linear. Therefore, the output of the functions will not be confined between any range.

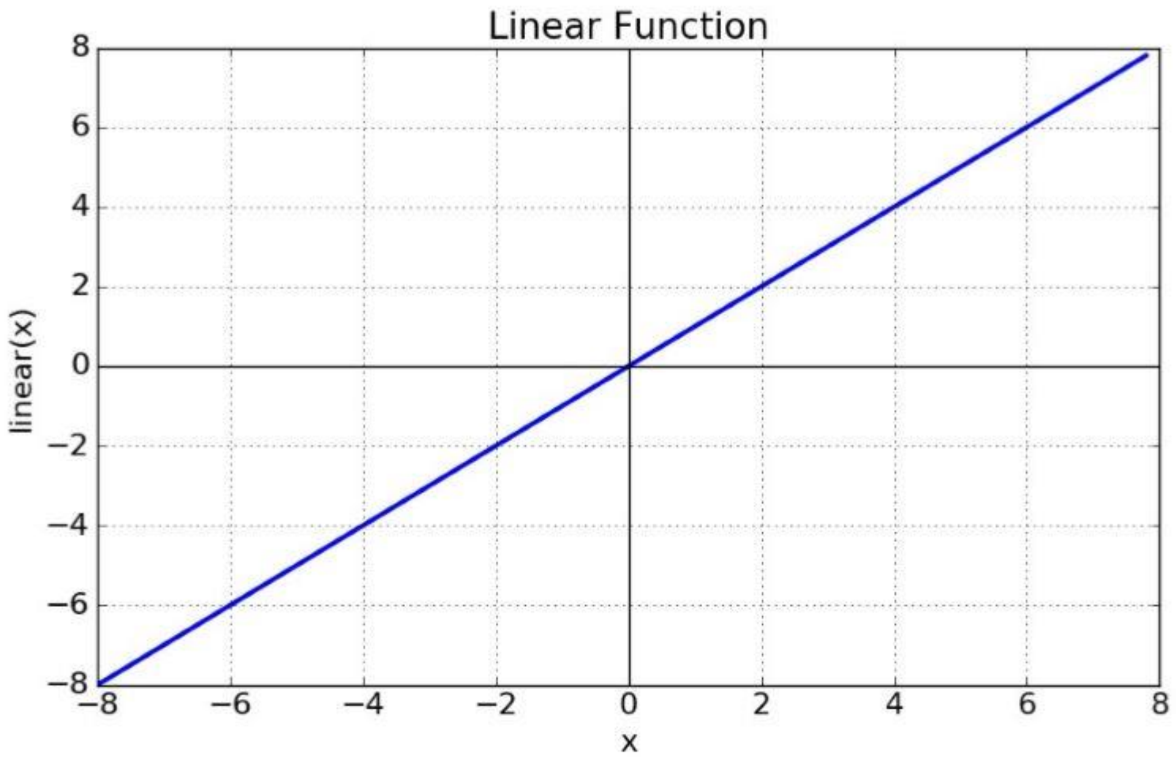


Fig: Linear Activation Function

Equation : f(x) = x

Range : (-infinity to infinity)

It doesn’t help with the complexity or various parameters of usual data that is fed to the neural networks.

Non-linear Activation Function

The Nonlinear Activation Functions are the most used activation functions. Nonlinearity helps to makes the graph look something like this

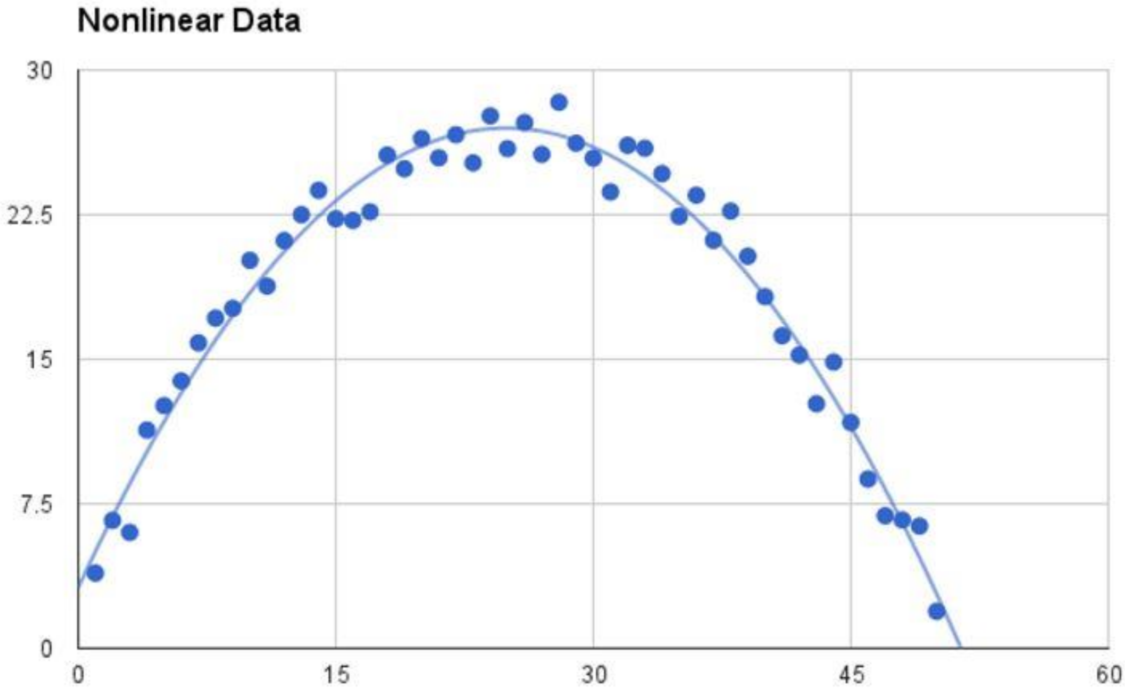


Fig: Non-linear Activation Function

It makes it easy for the model to generalize or adapt with variety of data and to differentiate between the output.

The main terminologies needed to understand for nonlinear functions are:

Derivative or Differential: Change in y-axis w.r.t. change in x-axis.It is also known as slope.
Monotonic function: A function which is either entirely non-increasing or nondecreasing.
The Nonlinear Activation Functions are mainly divided on the basis of their range or curves.

1. **Sigmoid or Logistic Activation Function**

The Sigmoid Function curve looks like a S-shape.

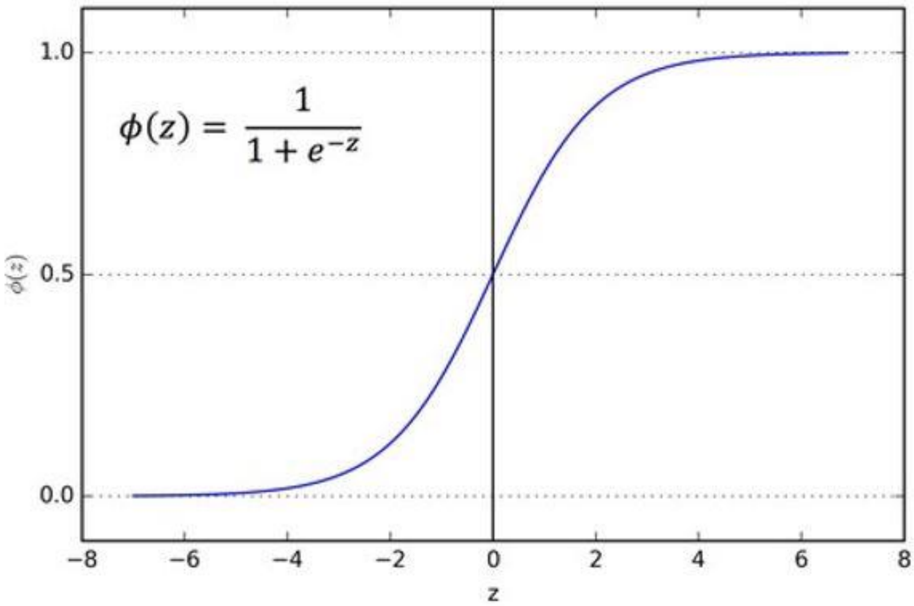


Fig: Sigmoid Function

The main reason why we use sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output.Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.

The function is differentiable. That means, we can find the slope of the sigmoid curve at any two points.

The function is monotonic but function’s derivative is not.
The logistic sigmoid function can cause a neural network to get stuck at the training time.
The softmax function is a more generalized logistic activation function which is used for multiclass classification.

2. **Tanh or hyperbolic tangent Activation Function**

Tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped).

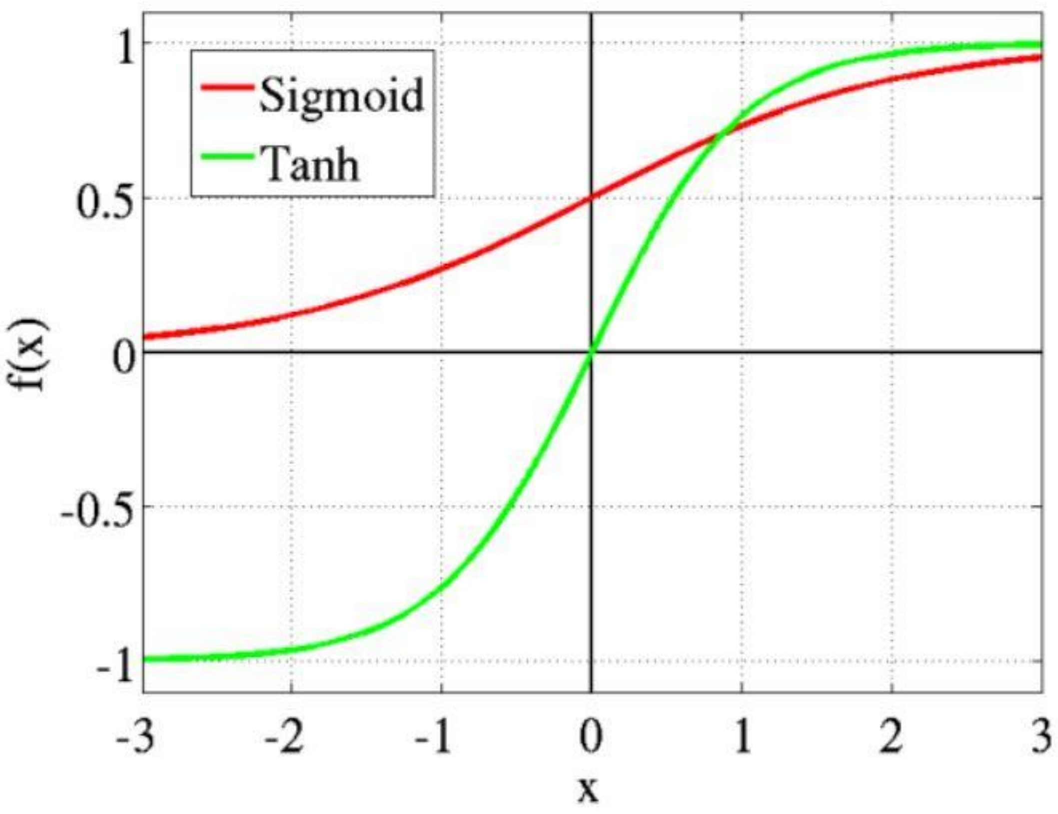


Fig: tanh v/s Logistic Sigmoid

The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.
The function is **differentiable**.
The function is **monotonic** while its **derivative is not monotonic**.
The tanh function is mainly used classification between two classes.
Note: *Both tanh and logistic sigmoid activation functions are used in feed-forward nets.*

3. ReLU (Rectified Linear Unit) Activation Function

The ReLU is the most used activation function in the world right now.Since, it is used in almost all the convolutional neural networks or deep learning.

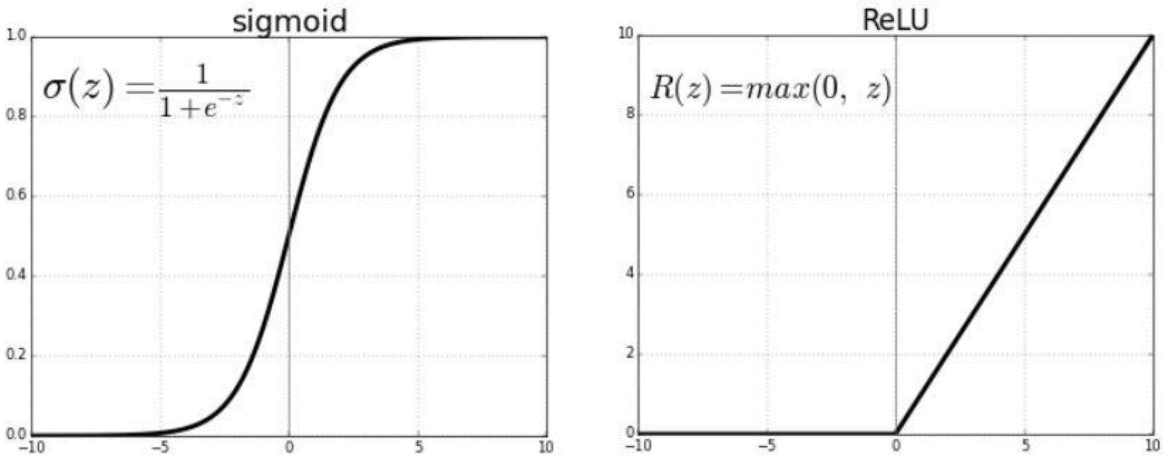


Fig: ReLU v/s Logistic Sigmoid

As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.
Range: [0 to infinity)
The function and its derivative **both are monotonic**.
But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.

4. **Leaky ReLU**

It is an attempt to solve the dying ReLU problem

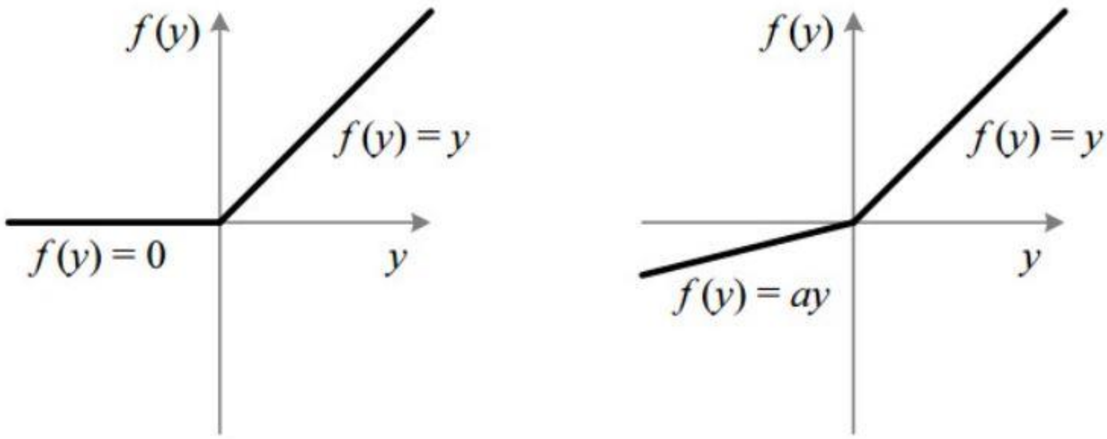


Fig : ReLU v/s Leaky ReLU

The leaky helps to increase the range of the ReLU function. Usually, the value of **a** is 0.01 or so.
When **a is not 0.01** then it is called **Randomized ReLU**.
Therefore the **range** of the Leaky ReLU is (-infinity to infinity).
Both Leaky and Randomized ReLU functions are monotonic in nature. Also, their derivatives also monotonic in nature.

Soft Computing

Soft Computing (SC) is an emerging area in Computer Science that is tolerant to imprecise and uncertain problems with partial truth to achieve an approximate, robust and low-cost optimal solution. The approach of SC techniques to solve problems imitate the remarkable power of human to think logically and learn from mistakes in an imprecise scenario. SC solves tremendous number of complex real world problems in different sections such as stock market predictions in business, computer aided diagnosis in medical, handwriting recognition in fraud detection, image retrieval, biometric application in image processing etc. and the list go on.

Hard Computing

The conventional algorithms, also termed as Hard Computing algorithms, follow mathematical methodologies strictly which make it inefficient to solve real world problems by taking more computation time. The conventional algorithms require exact input data, use a precise methodology and generate a precise output which make it a crisp system. It fails when the input is not exact. Examples of conventional algorithms are merge sort, quick sort, binary search, greedy algorithm, dynamic programming etc which are deterministic.

Branches of Soft Computing

Soft Computing consists of numerous techniques that study the biological processes such as reasoning, genetic evolution, survival of the creatures and human nervous system. SC is an umbrella term that thoroughly study the simulation of reasoning, human nervous system and evolution in different fields:

1. **Fuzzy Logic** is a technique that understands the vagueness of a solution and presents the solution with a degree of vagueness which is practical to human decision. It is widely applied in several applications of Artificial Intelligence for reasoning.
2. **Neural Network** is a network of artificial neurons, inspired by biological network of neurons, that uses mathematical models as information processing units to discover patterns in data which is too complex to notice by human.
3. **Evolutionary Computation** is a family of optimization algorithms that are inspired by biological evolution such as Genetic Algorithm, survival of creatures such as Particle Swarm Intelligence, Ant Colony Optimization, Artificial Bee Colony optimization etc. or any biological processes.

Soft Computing vs Hard Computing

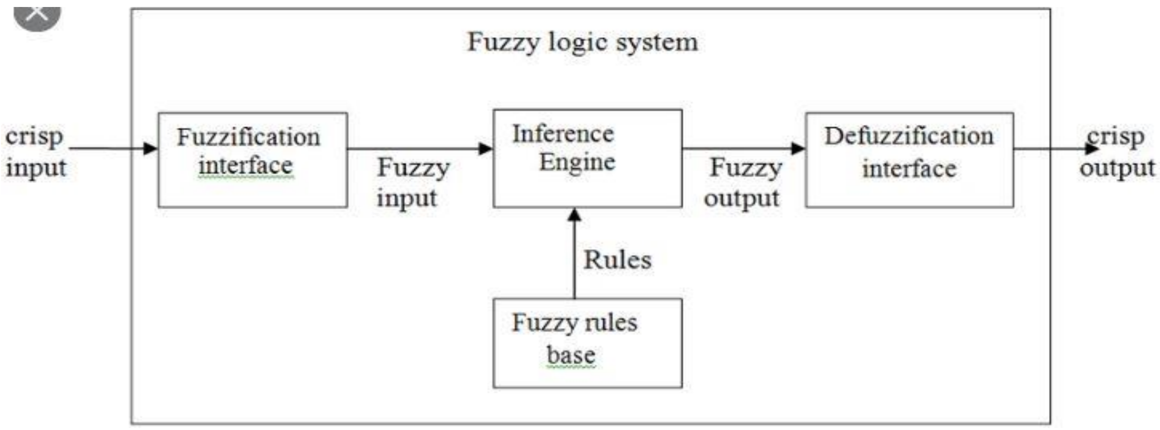
1. The biological processes fascinated scientists to solve real world problems by simulating the processes to robust algorithms and solve problems like a human mind in uncertain environment with limited information whereas the conventional algorithms (hard computing) fail to solve due to the strict principles. For example, conventional algorithms fail when input is not known/exact whereas SC deals with inexact information and generate a nearly optimal solution for the problem.
2. The conventional algorithms strictly follow a specific set of known steps to solve a task whereas SC techniques are heuristics.
3. Example: If we need to find out whether Bob is honest. A hard computing algorithm would give an answer that is YES or No. (1 or 0 in binary) whereas a SC technique (Fuzzy Logic) would give an answer with membership degree such as extremely honest

(1), very honest (0.85), sometimes honest (0.35), extremely dishonest (0.00), like a human.
4. Prediction Problem: Will tomorrow rain? Hard Computing would solve the problem by using statistical techniques which basically rely on user-driven hypothesis test that user defines variables, functions and type of interaction. These types of information can be manipulated and influence models. Whereas the hypothesis of Soft Computing techniques would scan all the predictor variables automatically and identify some patterns in the data that help to come up with accurate predictions. Such models have no chance to miss a notice of unexpected and potentially important variables, thereby lead to better accuracy.

Neuro-Fuzzy

Fuzzy Logic (FL) is a form of many-valued logic which deals with reasoning that is approximate rather than fixed and exact. Compared to traditional binary sets (where variables may take on true or false values), fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Due to the flexibility of FL concept, Fuzzy Logic Systems (FLSs) have attracted growing interest in modern information technology, production technique, decision making, pattern recognition, data mining, and medical diagnosis among others. FL has found a variety of applications in industrial process control and securities trading.

A typical FLS is strongly based on the concepts of fuzzy sets, linguistic variables and approximate reasoning. The fuzzifier transforms crisp inputs into fuzzy values while the Fuzzy Rule Base makes up the Knowledge Base which stores relevant data and knowledge of human experts in a specific domain such as the Decision-making unit combines all the fired rules for a given case and makes inference, while the defuzzifier converts fuzzy results into a crisp value for easy analysis and interpretations. Generally, when a problem has dynamic behaviour and involves several variables, FL technique can be applied to solve such problem. However, a major problem of the FLSs is the determination of its fuzzy sets and fuzzy rules which require deep knowledge of human experts in a particular domain. The Membership Functions (MFs) of FLSs are arbitrarily chosen, therefore fixed in nature. Generally, the shape of the MFs depends on certain parameters that can be adjusted. Rather than choosing the MF parameters arbitrarily, the neural network learning and tuning techniques provides a method for the FLS to learn information about a given dataset in order to automatically compute its MF parameters.



Basic architecture of a fuzzy logic system

Genetic algorithm

Genetic Algorithm (GA) is a search-based optimization technique based on the principles of **Genetics and Natural Selection**. It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve. It is frequently used to solve optimization problems, in research, and in machine learning.

Introduction to Optimization

Optimization is the process of **making something better**. In any process, we have a set of inputs and a set of outputs as shown in the following figure.

Optimization refers to finding the values of inputs in such a way that we get the "best" output values. The definition of "best" varies from problem to problem, but in mathematical terms, it refers to maximizing or minimizing one or more objective functions, by varying the input parameters.

The set of all possible solutions or values which the inputs can take make up the search space. In this search space, lies a point or a set of points which gives the optimal solution. The aim of optimization is to find that point or set of points in the search space.

What are Genetic Algorithms?

Nature has always been a great source of inspiration to all mankind. Genetic Algorithms (GAs) are search based algorithms based on the concepts of natural selection and genetics. GAs is a subset of a much larger branch of computation known as **Evolutionary Computation**.

In GAs, we have a **pool or a population of possible solutions** to the given problem. These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations. Each individual (or candidate solution) is assigned a fitness value (based on its objective function value) and the fitter individuals are given a higher chance to mate and yield more "fitter" individuals. This is in line with the Darwinian Theory of "Survival of the Fittest".

In this way we keep "evolving" better individuals or solutions over generations, till we reach a stopping criterion.

Genetic Algorithms are sufficiently randomized in nature, but they perform much better than random local search (in which we just try various random solutions, keeping track of the best so far), as they exploit historical information as well.

Advantages of GAs

GAs have various advantages which have made them immensely popular. These include −

- Does not require any derivative information (which may not be available for many real-world problems).

- Is faster and more efficient as compared to the traditional methods.
- Has very good parallel capabilities.
- Optimizes both continuous and discrete functions and also multi-objective problems.
- Provides a list of “good” solutions and not just a single solution.
- Always gets an answer to the problem, which gets better over the time.
- Useful when the search space is very large and there are a large number of parameters involved.

Limitations of GAs

Like any technique, GAs also suffer from a few limitations. These include −

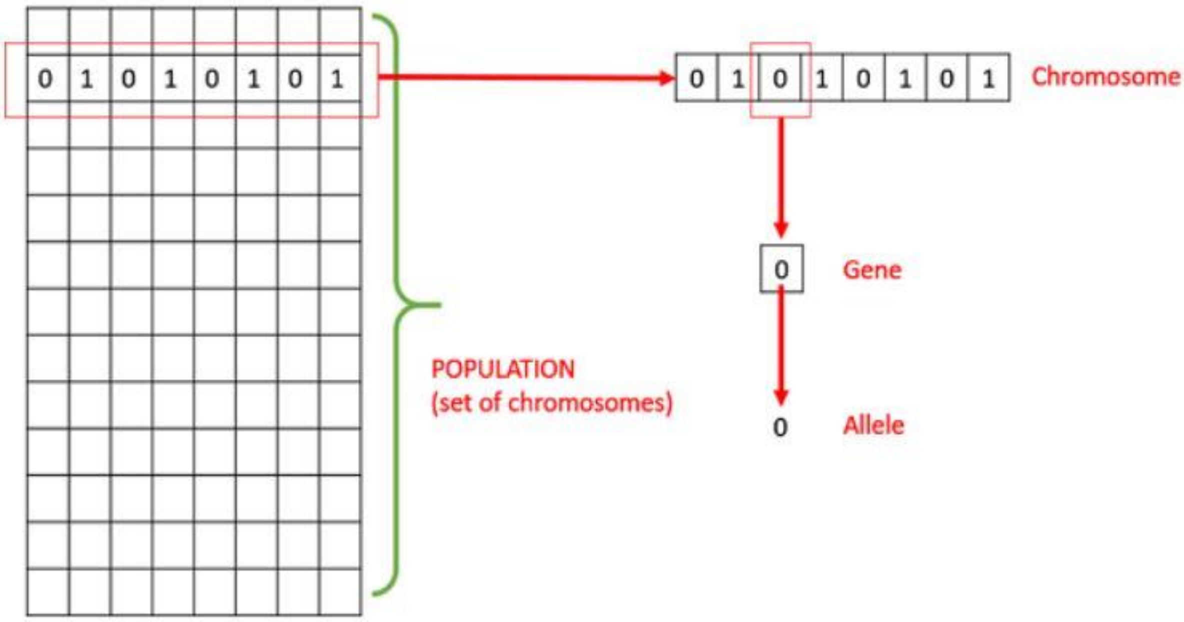
- GAs are not suited for all problems, especially problems which are simple and for which derivative information is available.
- Fitness value is calculated repeatedly which might be computationally expensive for some problems.
- Being stochastic, there are no guarantees on the optimality or the quality of the solution.
- If not implemented properly, the GA may not converge to the optimal solution.

Genetic Algorithm fundamental

Basic Terminology

Before beginning a discussion on Genetic Algorithms, it is essential to be familiar with some basic terminology which will be used throughout this tutorial.

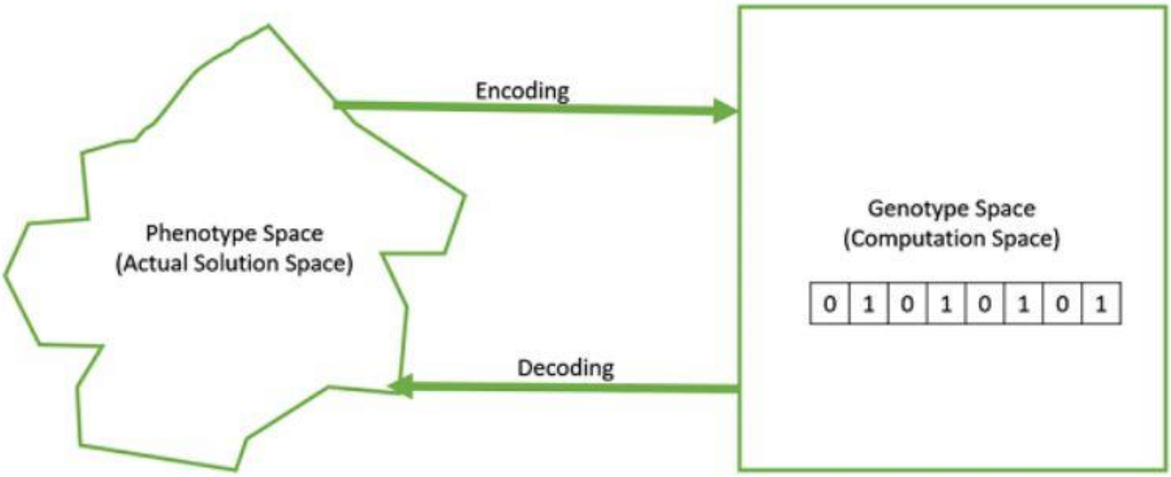
- **Population** − It is a subset of all the possible (encoded) solutions to the given problem. The population for a GA is analogous to the population for human beings except that instead of human beings, we have Candidate Solutions representing human beings.
- **Chromosomes** − A chromosome is one such solution to the given problem.
- **Gene** − A gene is one element position of a chromosome.
- **Allele** − It is the value a gene takes for a particular chromosome.



- **Genotype** − Genotype is the population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.
- **Phenotype** − Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.
- **Decoding and Encoding** − For simple problems, the **phenotype and genotype** spaces are the same. However, in most of the cases, the phenotype and genotype spaces are different. Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space. Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation.

  For example, consider the 0/1 Knapsack Problem. The Phenotype space consists of solutions which just contain the item numbers of the items to be picked.

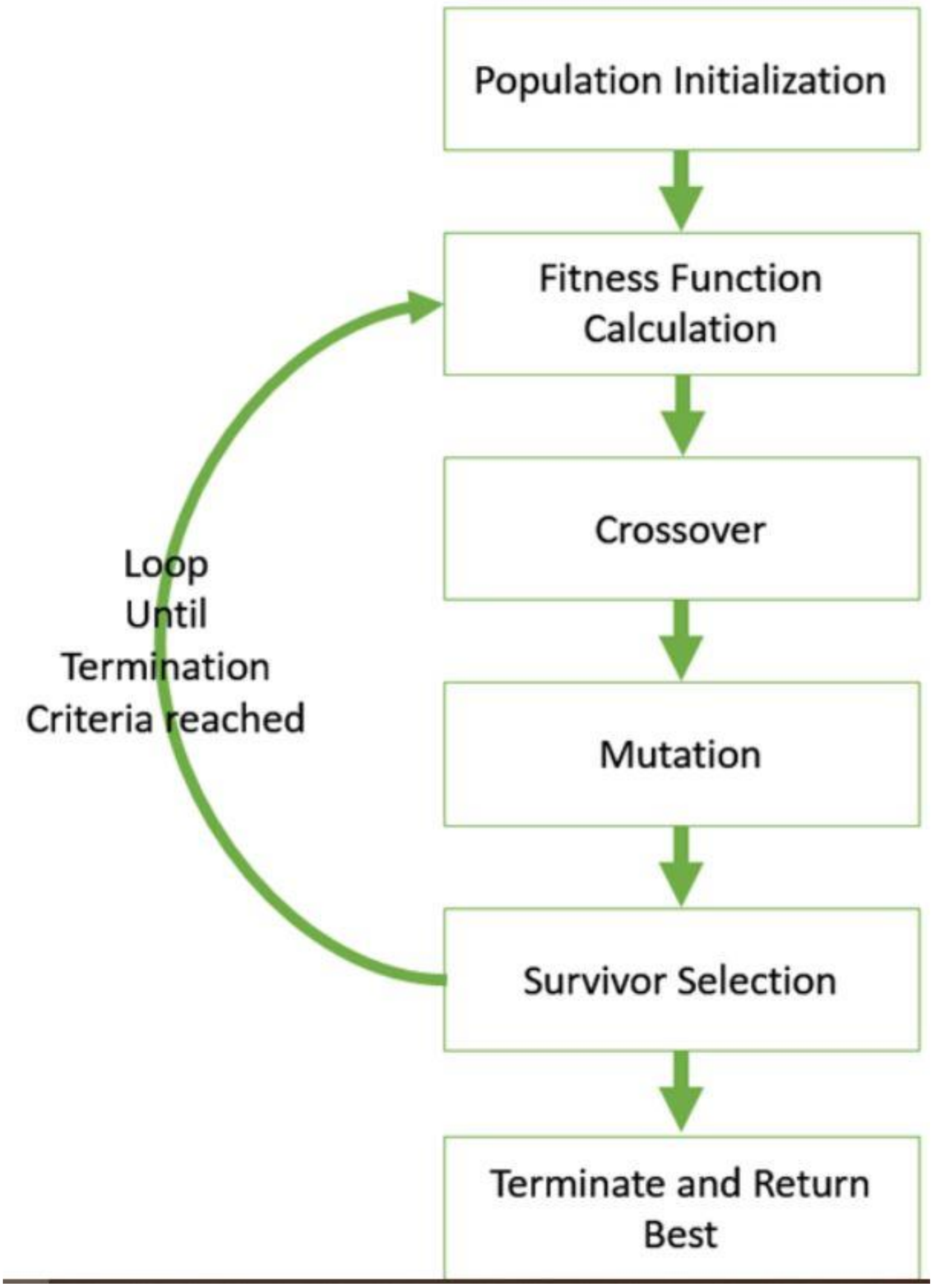
  However, in the genotype space it can be represented as a binary string of length n (where n is the number of items). A **0 at position x** represents that **x^th^** item is picked while a 1 represents the reverse. This is a case where genotype and phenotype spaces are different.



- **Fitness Function** − A fitness function simply defined is a function which takes the solution as input and produces the suitability of the solution as the output. In some cases, the fitness function and the objective function may be the same, while in others it might be different based on the problem.
- **Genetic Operators** − These alter the genetic composition of the offspring. These include crossover, mutation, selection, etc.

Basic Structure

We start with an initial population (which may be generated at random or seeded by other heuristics), select parents from this population for mating. Apply crossover and mutation operators on the parents to generate new off-springs. And finally, these off-springs replace the existing individuals in the population and the process repeats. In this way genetic algorithms actually try to mimic the human evolution to some extent.


Population Initialization
Fitness Function Calculation
Crossover
Mutation
Survivor Selection
Terminate and Return Best
Loop Until Termination Criteria reached