BALA GUGA GOPAL-S
1ST - MTECH .

NETWORK ARCHITECTURES :-

The manner in which the neurons of a neural network are structured is intimately linked with the learning algorithm used to train the network. We may therefore speak of learning algorithms rules based in the design of neural networks as being structured. The classification of learning algorithms is considered in section 3.

In general, we may identify three fundamentally different classes of network architectures:

i. Single-Layer feedforward Networks :-

In a layered neural network, the neurons are organized in the form of layers. In the simplest form of a layered network, we have an input layer of source nodes that projects onto an ~~layered network~~ output layer of neurons (computation nodes); but not vice versa. In other words, this network is strictly of a feedforward type. It is illustrated in fig box the case of four nodes in both the input & output layers. Such a network is called a single-layered network, with the designation "single-layer" referring to the output layer of computation nodes (neurons). We do not count the input layer of source nodes because no computation is performed there.

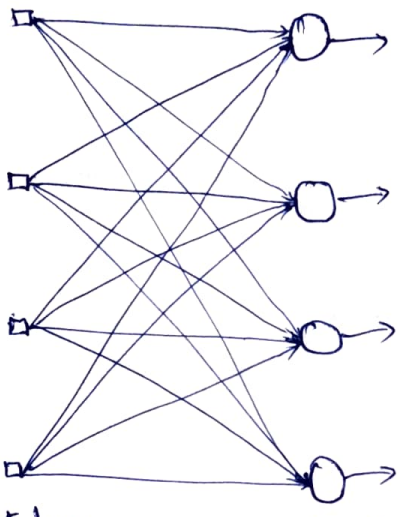
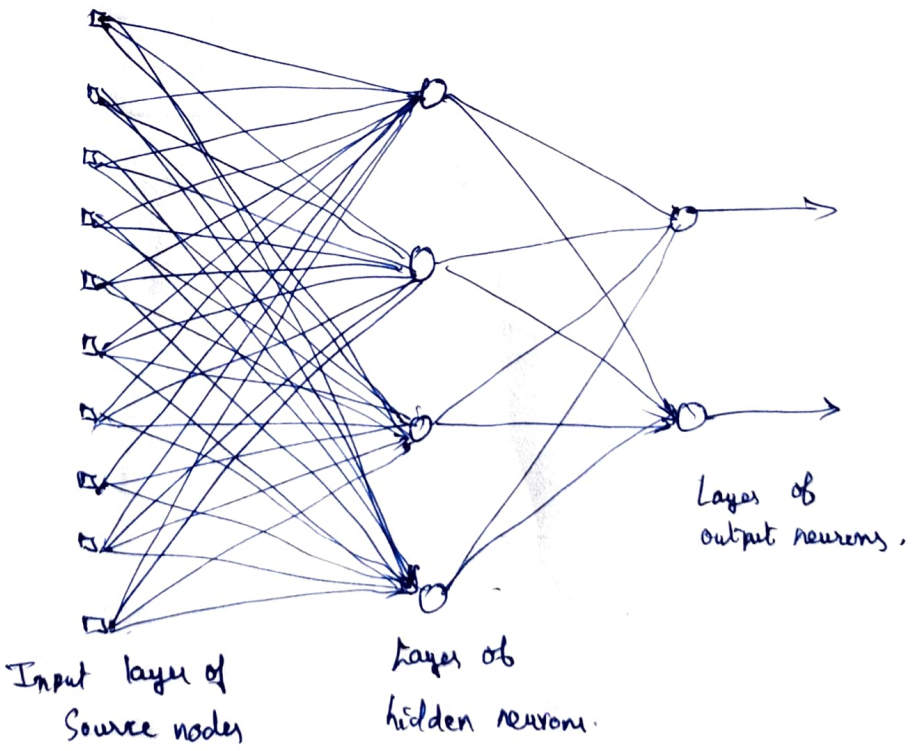


Fig - Feedforward network with a single layer of neurons :-

ii. Multilayer feedforward Networks.

The second class of a feedback forward neural network distinguishes itself by the presence of one or more hidden layers, whose computation nodes are correspondingly called hidden neurons or hidden units. The term hidden refers to the fact that this part of the neural network is not seen directly from either the input or output of the network. The function of hidden neurons is to intervene between the external input and the network's enable output in some useful manner. By adding one or more hidden layer, the network is enabled to extract higher-orders. Statistics from its input. In a rather loose sense, the network acquires a global perspective despite its local connectivity, due to the extra set of synaptic connections and the extra dimension of neural interactions.

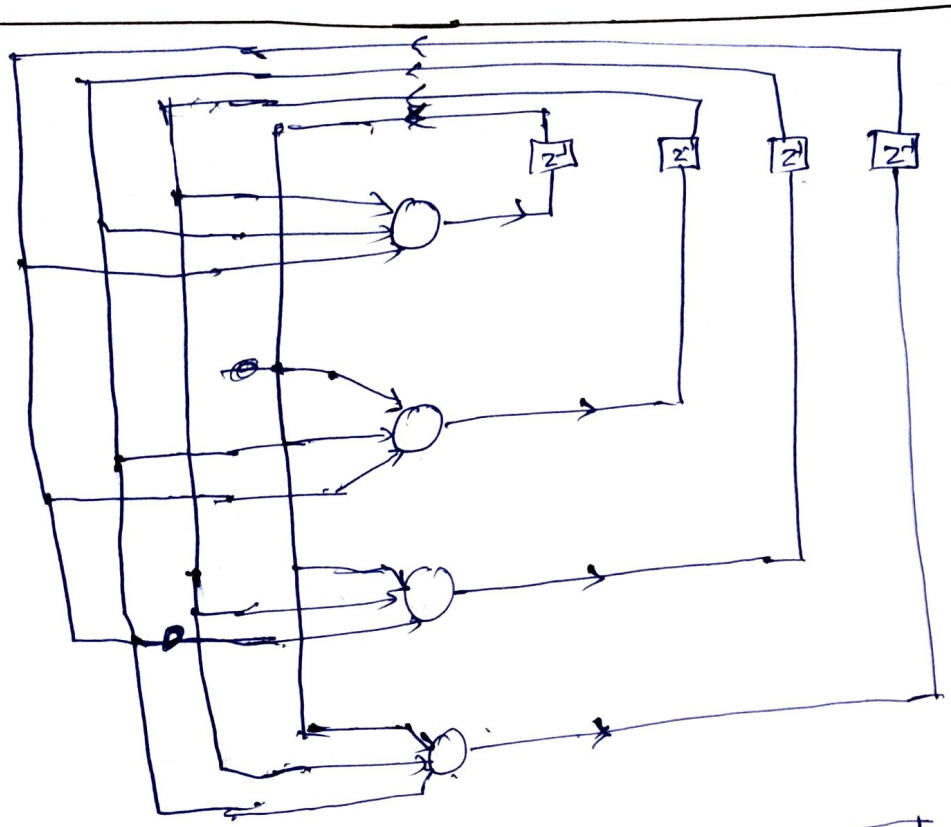
The Source node in the layer Input layer of the network supply respective elements of the activation pattern (which constitute the input signals applied to the neurons [computation nodes] in the second layer (i.e the first hidden layer). The output signals of the preceding layer only. The set of output signals of the neurons in the output layer of the network, constitutes the overall response of the network to the activation pattern supplied by the source nodes in the input [first] layer. The architectural graph in Fig 18 illustrates the layout of a multilayer feedforward neural network for the case of a single hidden layer. Eg. a feedforward network with m source nodes, h_1 neurons in the first hidden layer, h_2 neurons in the second hidden layer, and q neurons in the output layer is referred to as an $m-h_1-h_2-q$ network.



iii) Recurrent networks.

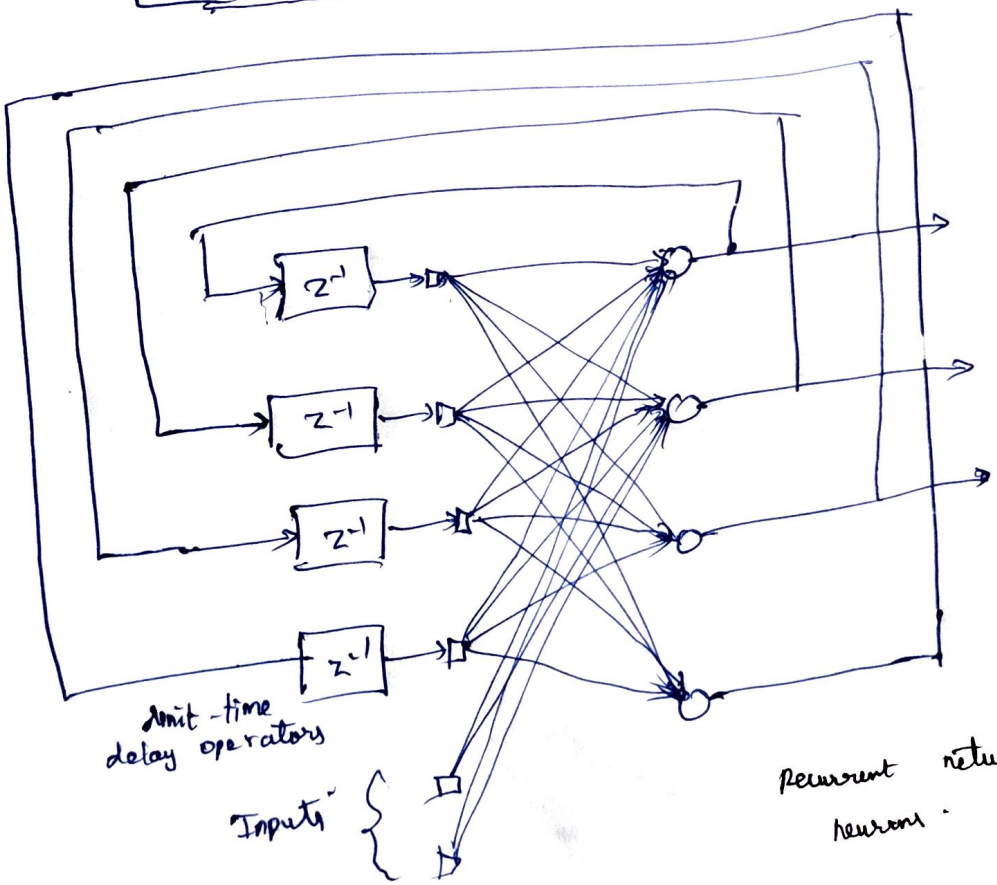
A recurrent neural network distinguishes itself from a feedforward neural network in that it has at least one feedback loop. For Eg, a recurrent network may consist of a single layer of neurons, with each neuron feeding its output signal back to the inputs of all the other neurons, as illustrated in the architectural graph. In the structure described in this fig, there are no self-feedback loops in the network. Self-feedback refers to a situation where the output of a neuron is fed back into its own input. The recurrent network illustrated in Fig, also has no hidden neurons.

The presence of feedback loop, be it in the recurrent structure of Fig or in that of Fig. has a profound impact on the learning capability of the network and on its performance. Moreover the feedback loops involve the use of particular branches composed of unit-time delay elements (denoted by z^{-1}), which result in a non-linear dynamic behaviour, assuming that the neural network contains non-linear units.



unit-time delay operators.

Fig. recurrent network with no. self-feedback loops and no hidden neurons.



Recurrent network with hidden neurons.

ACTIVATION FUNCTION:-

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

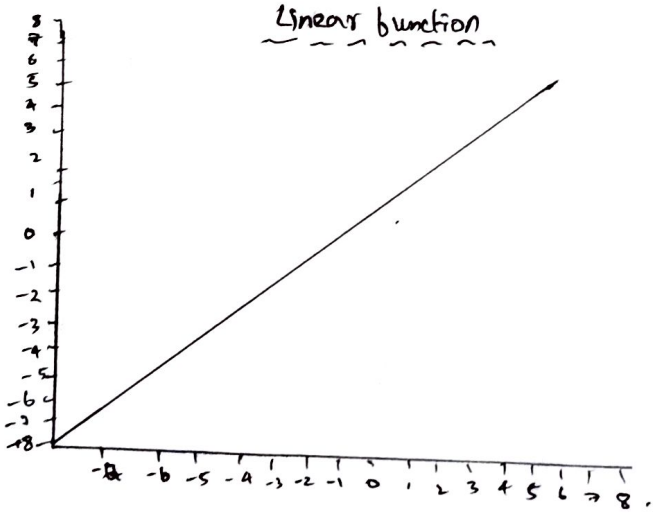
Activation functions are used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc.

The activation functions can be basically divided into 2 types.

1. Linear Activation function.
2. Non-linear Activation functions.

linear or Identity Activation Function

The output of the functions will not be confined between any range.



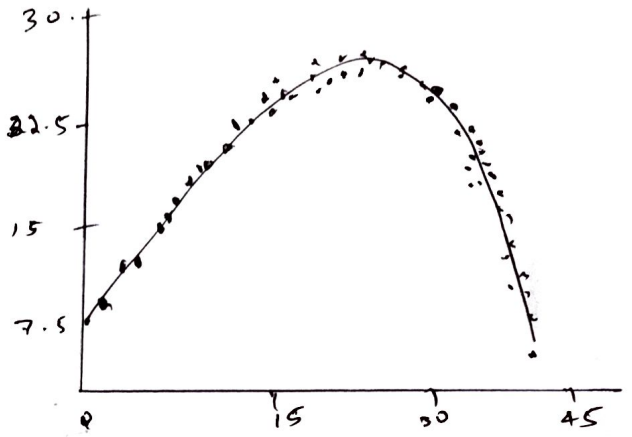
eq: $f(x) = x$.

Range → $(-\infty$ to $\infty)$

It does not help with complexity of various parameters of usual data that is feed to the neural networks.

Non-linear Activation function :-

The non-linear activation functions are the most used activation functions, non-linearity helps to make the graph look something like this.

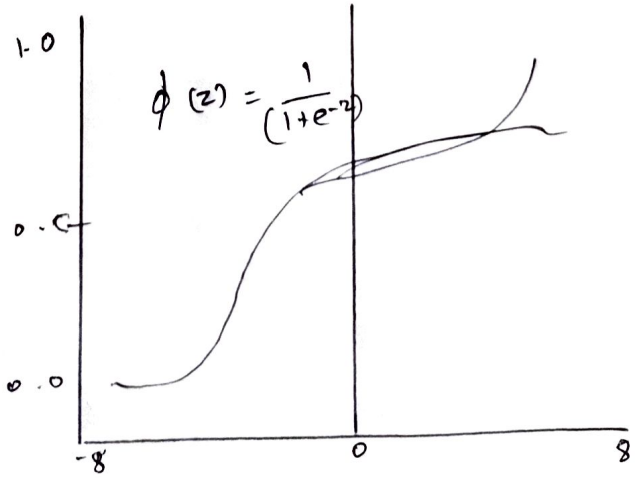


It makes it easy for the model to generalize or adapt with variety of data and to differentiate between the output.

The non-linear activation functions are mainly divided on the basis of their range or curves.

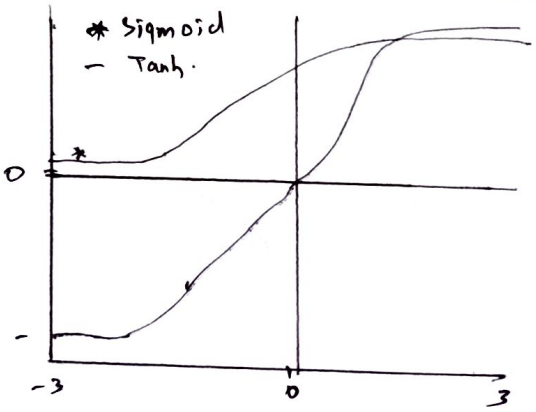
Sigmoid or logistic Activation function

The Sigmoid function curve looks like a S-shape



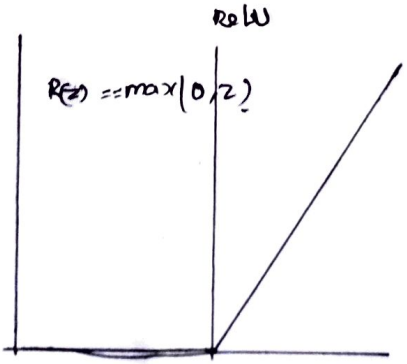
The main reason why we use Sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 & 1, Sigmoid is the right choice, we can find the slope of the Sigmoid curve at any two points, the function is monotonic but functions derivative is not.

b. TANh or hyperbolic tangent Activation Function.



The tanh function is mainly used classification between two classes.

c. Relu (Rectified linear unit) Activation function:-



As it is visible, the Relu is half rectified (from bottom) f(z) is zero, when z is less than zero & f(z) is equal to z when z is above or equal to zero

Range (0 - ∞).

BALA GUGA GOPAL S.
1st MTech.

LEARNING PROCESS.

There are different ways in which we ourselves learn from our Surroundings. Same goes with neural networks also. In a board sense, we may categorize the learning process through which neural networks function as follows: LEARNING WITH TEACHER & LEARNING WITHOUT TEACHER. The Learning without a teacher can be further subcategorised into unsupervised learning and reinforcement learning.

i. Learning with teacher

Learning with a teacher is also referred as Supervised Learning. In Conceptual terms, we may think of the teacher as having knowledge of the environment, with that knowledge being represented by a set of input-output examples. The environment is however, unknown to the neural network. Suppose now that the teacher and the neural network are both exposed to a training vector [i.e. example] drawn from the same environment. By the virtue of built-in knowledge, the teacher is able to provide the neural network with a desired response for that training vector. Indeed, the desired response represents the "optimum" action to be performed by the neural network. The network parameters are adjusted under the combined influence of the training vector and the error signal. The error signal is defined as the difference between the desired response and the actual response of the network.

This adjustment is carried out iteratively in a step-by-step fashion with the aim of eventually making the neural network emulate the teacher, the emulation is presumed to be optimum in some statistical sense. In this way, knowledge of the environment available to the teacher is transferred to the neural network through training and transferred to the neural stored in the form of "fixed" synaptic weights, representing long-term memory. When this condition is reached, we may then dispense with the teacher and let the neural network deal with the environment completely by itself.

The form of supervised learning we have just described is the basis of error-correction learning. The supervised learning process constitutes a closed-loop feedback system, but the unknown environment is outside the loop. As a performance measure for the system, we may think in terms of the mean square error, or the sum of squared errors over the training sample, defined as a function of the free parameters of the system.

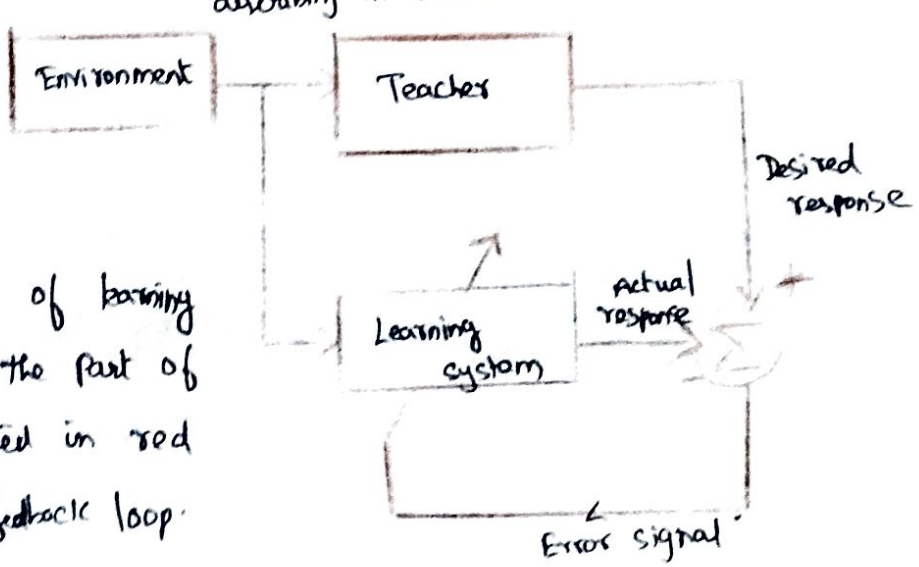


Fig :- Block diagram of learning with a teacher, the part of the figure printed in red constitutes a feedback loop.

This function may be visualised as a multidimensional error-performance surface, or simply error surface with the free parameters as coordinates. The error surface is averaged over all possible input-output examples. Any given operation of the system under the teacher's supervision is represented as a point on the error surface. For the system to improve performance over time and therefore learn from the teacher, the operating point has to move down successively toward a minimum point of the error surface, the minimum point may be a local minimum or a global minimum. A supervised learning system is able to do this the useful information it has about the gradient of the error surface corresponding to the current behaviour of the system.

The gradient of the error surface at any point is a vector that points in the direction of steepest decent. In fact, in the case of supervised learning from examples, the system may use an instantaneous estimate of the gradient vector, with the example indices presumed to be those of time. The use of such an estimate results in a motion of the operating point on the error surface that is typically in the form of a random walk. Nevertheless, given an algorithm designed to minimise the cost function, an adequate set of input-output examples, and enough time in which to do the training, a supervised learning system is usually able to approximate an unknown input-output mapping reasonably well.

LEARNING WITHOUT A TEACHER.

In supervised learning process takes place under the tutelage of a teacher. As the name implies, here there is no teacher to oversee the learning process. That is to say, there are no labled examples of the function to be learned by the network. Under this second paradigm, two subcategories are identified.

1. Reinforcement learning

In reinforcement learning, the learning of an input-output mapping is performed through continued interaction with the environment in order to minimize a scalar index of performance. Fig shows the block diagram of one form of a reinforcement learning system that converts a primary reinforcement signal received from the environment into a higher quality reinforcement signal called the heuristic reinforcement signal, both of which are scalar inputs. The system is designed to learn under delayed reinforcement, which means that the system observes a temporal sequence of stimuli also received from the environment, which eventually results in the generation of the heuristic reinforcement signal.

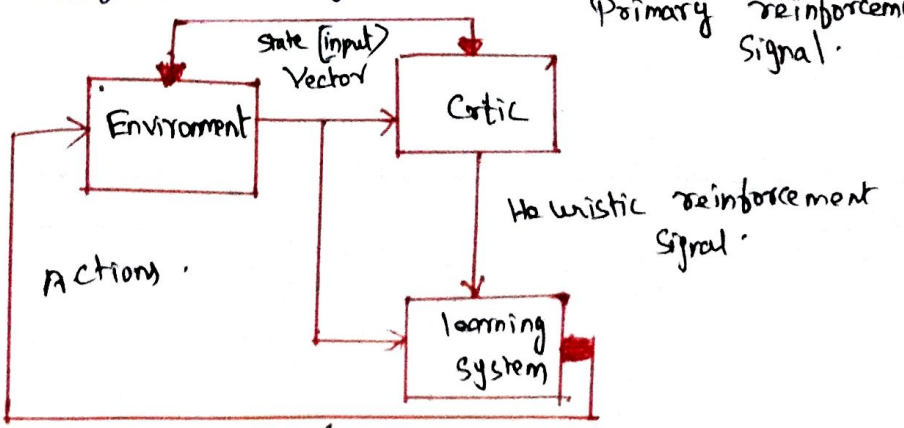


Fig- Block diagram of reinforcement learning, the learning system & the environment are both inside the feedback loop.

The goal of reinforcement learning is to minimize a cost-to-go function, defined as the expectation of the cumulative cost of actions taken over a sequence of steps instead of simply the immediate cost. It may turn out that certain actions taken earlier in that sequence of time steps are in fact the best determinants of overall system behaviour. The function of the learning system is to discover these actions and feed them back to the environment.

Delayed-reinforcement learning is difficult to perform for two basic reasons:

* There is no teacher to provide a desired response at each step of the learning process
* The delay incurred in the generation of the primary reinforcement signal implies that the learning machine must solve a temporal credit assignment problem. By this we mean that the learning machine must be able to assign credit & blame individually to each action in the sequence of time steps that led to the final outcome, while the primary reinforcement may only evaluate the outcome.

Notwithstanding these difficulties, delayed-reinforcement learning is appealing. It provides the basis for the learning system to interact with its environment, thereby developing the ability to learn to perform a prescribed task solely on the basis of the outcomes of its experience that results from the interaction.

2. unsupervised learning.

In unsupervised or self-organised learning, there is no external teacher or critic to oversee the learning process, as indicated in Fig. Rather, provision is made for a task-independent measure of the quality of representation that the network is required to learn, and the free parameters of the network are optimized with respect to that measure.

To perform unsupervised learning, we may use a competitive-learning rule. For example, we may use a neural network that consist of two players- an input layer & a competitive layer. The input layer receives the available data. The competitive layer consists of neurons that compete with each other for the opportunity to respond to features contained in the input data. In its simplest form, the network operates in accordance with a "winner-take-all" strategy. In such a strategy, the neuron with the greatest total input "wins" the competition and turns on. All the other neurons in the network then switch off.

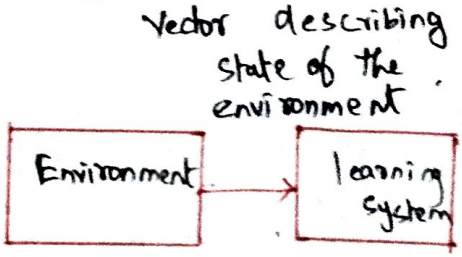


Fig. Block diagram of unsupervised learning.

LEARNING TASKS

The choice of a particular learning rule, is of course, influenced by the learning task, the diverse nature of which is testimony to the universality of neural networks.

Pattern association

An associative memory is a brainlike distributed memory that learns by association. Association has been known to be prominent feature of human memory since the time of Aristotle, and all models of cognition use association in one form or another as the basic operation (Anderson, 1995).

Association takes one of two forms, auto association and heteroassociation. In auto association, a neural network is required to store a set of patterns [vectors] by repeatedly presenting them to the network. ~~is required to store~~ The network is subsequently presented with a partial description or distored (noisy) version of an original pattern stored in it, and the task is to retrieve (recall) that particular pattern. Heteroassociation differes from auto association in that an arbitrary set of input patterns is paired with an other arbitrary set of output patterns. Auto association involves the use of unsupervised learning, where as the type of learning involved in hetero association is supervised.

Let x_k denote a key pattern [vector] applied to an associative memory and y_k denote a memorised pattern (vector). The pattern association performed by the network is described by

$$x_k \Rightarrow y_k \quad , \quad k = 1, 2, 3 \ldots q.$$

where q is the number of patterns stored in the network. The key pattern x_k acts as a stimulus that not only determines the storage location of memorized pattern y_k, but also holds the key for its retrieval.

In an autoassociative memory, $y_k = x_k$, So the input & output (data) spaces of the network have the same dimensionality. In a heteroassociative memory $y_k \neq x_k$, hence, the dimensionality of the output space in this second case may or may not equal the dimensionality of the input space.

There are two phases involved in the operation of an associative memory:

* Storage phase :- which refers to the training of the network in accordance with Eq. $\longrightarrow$

$$\Rightarrow x_k \rightarrow y_k \quad , \quad k = 1, 2, \cdots q.$$

* Recall phase :- which involves the retrival of a memorized pattern in response to the presentation of a noisy or distorted version of a key pattern to the network.

Let the stimulus (input) x represent a noisy or distorted version of a key pattern x_j. This stimulus produces a response (output) y, as indicated in Fig. For perfect recall, we should find that $y = y_j$, where y_j is the memorized pattern associated with the key pattern x_j. When $y \neq y_j$ for $x = x_j$, the associative memory is said to have made an error in recall.

Fig.

the number of patterns q stored in an associative memory provides a direct measure of the storage capacity of the network. In designing an associative memory, the challenge is to make the storage capacity q [expressed as a percentage of the total number N of neurons used to construct the network] as large as possible, yet insist that a large fraction of the memorized patterns is recalled correctly.

Pattern Recognition :-

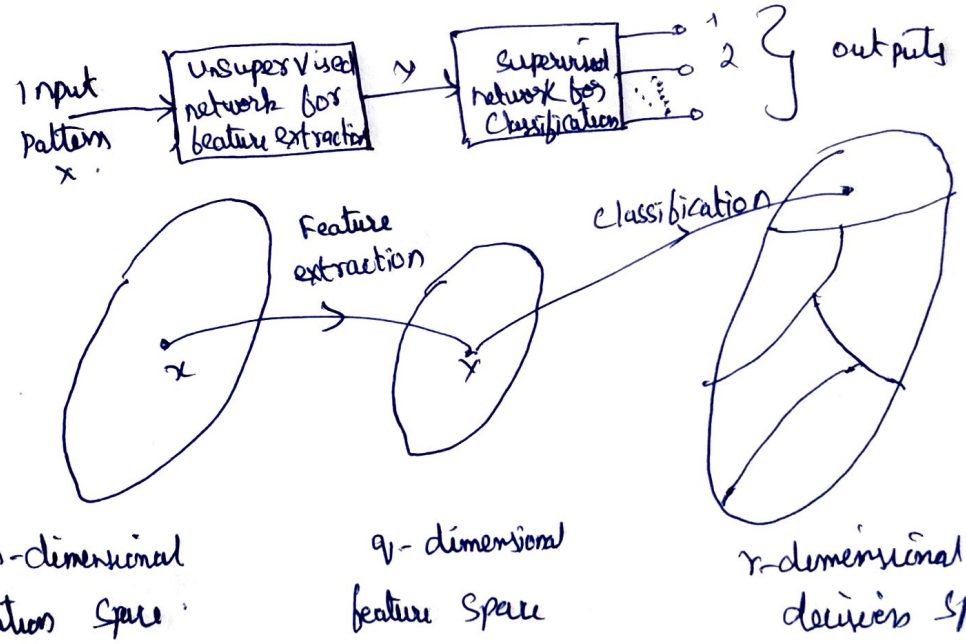
Humans are good at pattern recognition, we receive data from the world around us via our senses and are able to recognize the source of the data. we are often able to do so almost immediately and with practically no effort. For eg, we can recognize the familiar face of a person by his or her voice on the telephone despite a bad connection, and distinguish a boiled egg that is good from a bad one by smelling it. Humans perform pattern recognition through a learning process. So it is with neural networks.

Pattern recognition is formally defined as the process whereby a received pattern/signal is assigned to one of a prescribed number of classes. A neural network performs pattern recognition by first undergoing a training session during which the network is repeatedly presented with a set of input patterns along with the category to which each particular pattern belongs. Later, the network is presented with a new pattern that may not have been seen before. The decision space is divided into regions, each one of which is associated with a class. The decision

The decision boundaries are determined by the training process. The construction of these boundaries is made statistical by the inherent variability that exists within and between classes.

In generic terms, pattern-recognition machines using neural networks may take one of two forms.

* The machine is split into two parts, an unsupervised network for feature extraction and a supervised network for classification, as shown in the hybridized system of Fig. Such as method follows the traditional approach to statistical pattern recognition. In conceptual terms, a pattern is represented by a set of m observables, which may be viewed as a point x in an m-dimensional observation (data) space.



Feature extraction is described by a transformation that maps the point x into an intermediate point y in a q-dimensional feature space with $q < m$, as indicated in Fig. The transformation may be viewed as one of dimensionality reduction, the use of which is justified on the grounds that it simplifies the task of clas

* The machine is designed as a feedback network using a Supervised learning algorithm. In this second approach, the task of feature extraction is performed by the computational units in the hidden layer(s) of the network.

Function Approximation:

The third learning task of interest is that of function approximation. Consider a non-linear input-output mapping described by the functional relationship

$$d = f(x).$$

where the vector x is the input and the vector d is the output. The vector valued function $f(\cdot)$ is assumed to be unknown. To make up for the lack of knowledge about the function $f(\cdot)$, we are given the set of labeled examples.

$$\mathcal{A} = \{(x_i, d_i)\}_{i=1}^{N}.$$

The requirement is to design a NN that approximates the unknown function $f(\cdot)$ such that the function $F(\cdot)$ describing the input-output mapping actually realized by the network is close enough to $f(\cdot)$ in a Euclidean sense over all inputs, as shown by.

$$\|F(x) - f(x)\| < \varepsilon \quad \text{for all } x,$$

where ε is a small positive number. Provided that the the size N of the training sample $\mathcal{A}$ is large enough & the network is equipped with an adequate number of free parameters, then the approximation error ε can be made small enough for the task.

The approximation problem described here is a perfect candidate for supervised learning, with x_i playing the role of input vectors & d_i serving the role of desired response. we may turn this issue around & view supervised learning as an approximation problem.

The ability of a neural network to approximate an unknown input-output mapping may be exploited in two important ways:-

i- system identification:-

Describe the input-output relation of an unknown memoryless multiple input-multiple output (MIMO) system, by a memoryless system, we mean a system that is time invariant. We may then use the set of labeled examples in eq. to train a neural network as a model of the system. Let the vector y_i denote the actual output of the neural network produced in response to an input vector x_i. The difference between d_i (associated with x_i) & the network output y_i provides the error signal vector e_i.

ii. INVERSE MODELING:-

Suppose next we are given a known memoryless MIMO system whose input-output relation is described. The requirement in this case is to construct an inverse model that produces the vector x in response to the vector d. The inverse system may thus be described by $x = f^{-1}(d)$.

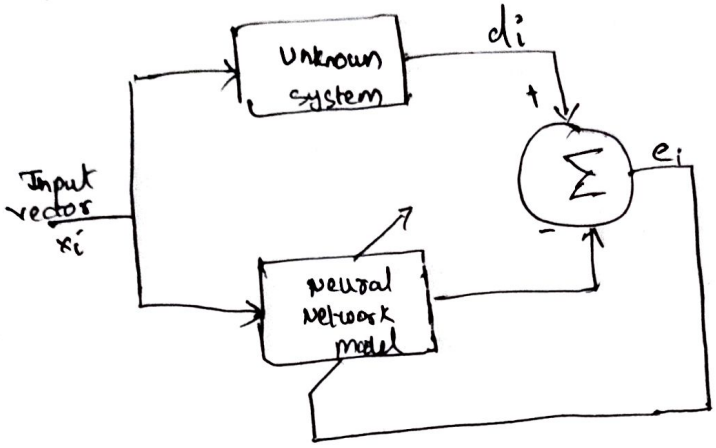


Fig :- Block diagram of system identification. The neural network, doing the identification is part of the feedback loop.

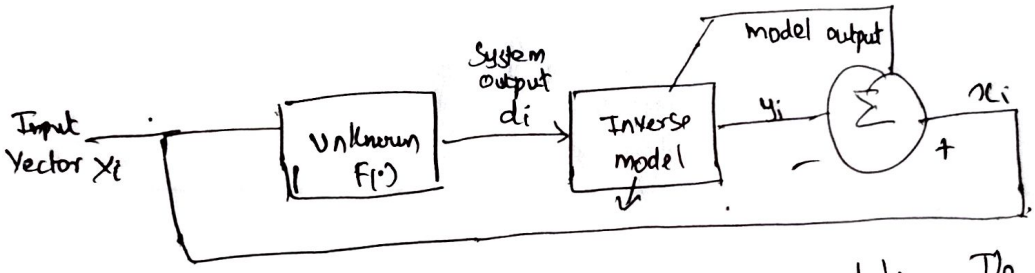


Fig:- Block diagram of inverse system modeling. The neural network acting as the inverse model, is part of the feedback loop.

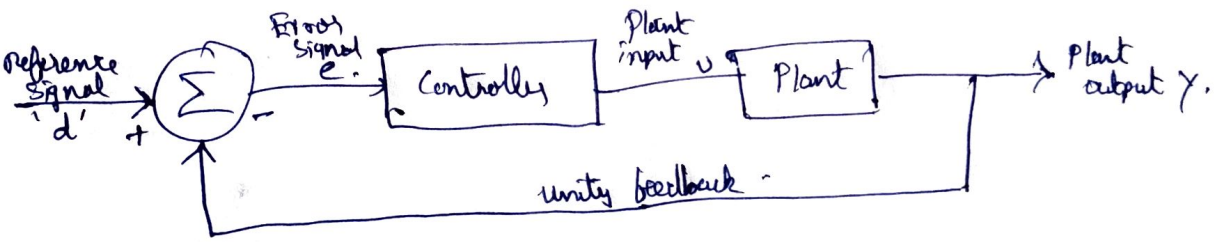
where the vector-valued function $F^{-1}(\cdot)$ denotes the inverse of $F(\cdot)$. Note, however, that $f^{-1}(\cdot)$ is not the reciprocal of $f(\cdot)$, rather, the use of superscript -1 is merely a flag to indicate an inverse. In many situations encountered in practice. The vector-valued function $F(\cdot)$ is much too complex and inhibits a straightforward formulation of the inverse function $f^{-1}(\cdot)$. Given the set of labled examples in Eq. we may construct a neural network approximation of $f^{-1}(\cdot)$ by using the scheme shown in Fig 30. In the situation described here, the roles of x_i & d_i are interchanged. The vector d_i is used as the input and x_i is treated as the desired response. Let the error signal vector e_i denote the difference between x_i & the actual output y_i of the neural network produced in response to d_i. As with the system identification problem.

this error signal vector is used to adjust the free parameters of the neural network, Produced in response to d_i. As with the system identification Problem, this error signal vector is used to adjust the free parameters of the neural network to minimize the squared difference between the outputs of the unknown inverse system and the neural network in a Statistical Sense, and is computed over the complete training set of J. Typically, inverse modeling is a more difficult learning task than System identification, as there may not be a unique Solution for it.

Control:-

The Control of a plant is another learning task is well suited for neural networks, by a plant we mean a process or critical part of a System that is to be maintained in a controlled condition. The relevance of learning to control should not be surprising because, after all, the human brain is a computer (i.e. information processor), the outputs of which as a whole system are actions. In the context of control, the brain is living proof that it is possible to build a generalized controller that takes full advantage of parallel distributed hardware, can control many thousands of actuators (muscle fibers) in parallel, can handle non-linearity and noise, and can optimize over a long-range planning horizon (werbos, 1992).

Consider the feedback Control System shown in Fig 3.1 . The System involves the use of unity feedback around a plant to be Controlled that is the plant output is fed back directly to the input. Thus, the plant output y is subtracted from a reference signal d supplied from an external source. The error signal e so produced is applied to a neural Controller for the purpose of adjusting its free parameters. The primary objective of the Controller is to supply appropriate inputs to the plant to make its output y track the reference signal d. In other words, the Controller has to invert the plant's input-output behaviour.



the error signal e has to propagate through the neural Controller before reaching the plant. Consequently, to perform adjustments on the free parameters of the plant in accordance with an error-correction learning algorithm, we need to know the Jacobian, made up of a matrix of partial derivatives as shown by.

$$J = \left\{ \frac{\partial y_k}{\partial u_j} \right\} \quad j, k.$$

where y_k is an element of the plant output y & u_j is an element of the plant input u. Unfortunately, the partial derivatives $\frac{\partial y_k}{\partial u_j}$ for the various k & j depend on the operating point of the plant and are therefore not known. We may use one of two approaches to account for them.

i) Indirect learning :- Using actual input-output measurements on the plant we first construct a neural model to produce a copy of it. This model is in turn, used to provide an estimate of the Jacobian. The partial derivatives constituting this Jacobian are subsequently used in the error-correction learning algorithm for computing the adjustments to the free parameters of the neural controller. [Nguyen & Widrow 1989, Suyhen et al. 1996, Widrow & Walach 1996]

ii) Direct learning :- The signs of the partial derivatives $\partial y_k / \partial u_i$ are generally known and usually remain constant over the dynamic range of the plant. This suggests that we may approximate these partial derivatives by their individual signs. These absolute values are given a distributed representation in the free parameters of the neural controller. The neural controller is thereby enabled to learn the adjustments to its free parameters directly from the plant.

Beamforming :-

Beamforming is used to distinguish between the spatial properties of a target signal & background noise. The device used to do the beamforming is called a beamformer.

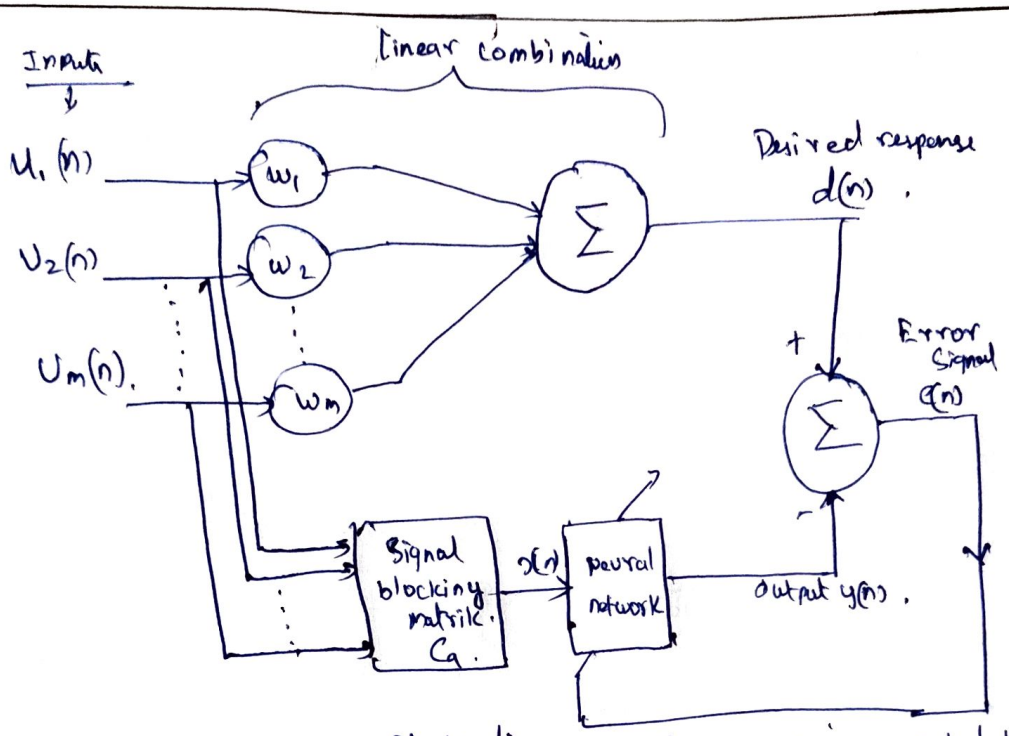
The task of beamforming is computable, for example, with feature mapping in the cortical layers of auditory systems of echolocating bats. The echolocating bat illuminates the surrounding environment by broadcasting short duration freq-modulated [FM] sonar signals and then uses its auditory system [including a pair of ears] to focus

Beamforming is commonly used in radar & Sonar systems, where the primary task is to detect and track a target of interest in the combined presence of receiver noise & interfering signals (e.g jammers). This task is complicated by 2 factors.

* The target signal originates from an unknown direction, &
* There is no prior information available on the interfering signals.

One way of coping with situation of this kind is to use a generalized sidelobe canceller (GSLC), the block diagram of which is shown in fig. The system consists of the following components.

* An array of antenna elements, which provides a means of sampling the observation space signal at discrete points in space.
* A linear combiner defined by a set of fixed weights $(w_i)_{i=1}^{m}$, the output of which performs the role of a desired response. This linear combiner acts like a 'spatial filter', characterized by a radiation pattern.
* A signal-blocking matrix C_a, the function of which is to cancel interference that leaks through the sidelobes of the radiation pattern of the spatial filter represented the linear combiner.



Block diagram of generalized sidelobe Canceller.

* A neural network with adjustable parameters, which is designed to accomodate statistical variations in the interfering signals.

The adjustments to the free parameters of the neural network are performed by an error-correcting learning algorithm that operates on the error signal $e(n)$, defined as the difference between the linear combiner output $d(n)$ and the actual output $y(n)$ of the neural network. Thus the GSLC operates under the supervision of the linear combiner that assumes the role of a "teacher". As with ordinary supervised learning, notice that the linear combiner is outside the feedback loop acting on the neural network. A beamformer that uses a neural network for learning is called a neuro-beamformer. This class of learning machine comes under the general heading of attentional neurocomputers.