UNIT 4

OPTICAL NEURAL NETWORKS, THE COGNITRONS AND NEOCOGNITRONS AND GENETIC ALGORITHMS

4.1 OPTICAL NEURAL NETWORKS –

Q.1 Explain Optical Neural Networks. **(KUK, May 2009 & May 2010)**

Q.2 What is an optical Neural Networks? Explain its type. **(KUK, June 2010)**

Ans. Optical Neural netwroks intercorrect neurons with light beams.As a result no insulation is required between signal paths,v the light rays can pass through between each other without interacting. The signal path can be made to travel in three dimensions. The density of the transmission path is limited only by the spacing of light sources, the effect of divergence and the spacing of the detectors. As a result all signal paths operate simultaneously, which result in a true data rate. The strengths of weight are stored in holograms with high density. These weights can be modified during operation to produce a fully adaptive system.

Optical neural networks promise a way. By interconnecting neurons with light beams, no insulation is required b/w signal paths, the light rays can pass through each other without interacting, the signal path can be made in three dimensions. Also the transmission path density is limited only by spacing of light sources. Optical nrural network can also provide important computational benefits. Given all of these advantages, one might be expected to ask why anyone would make a network in any other way. Unfortunately there are many problems related with optical implemantations. Optical devices have their own physical characterstics & they often do not match with the requirement of neural networks. There are two categories of this optical neural networks. They are electro-optical matrix multiplier and Holographic correlators.

4.2 VECTOR MATRIX MULTIPLIER –

Q.3 Explain Vector Matrix Multiplier. **(KUK, May 2010)**

Ans. The application of most artifical neural network can be described mathematically as a series of vector matrix multiplication one for each layer.

To calculate the output of each layer, input vector is applied and is multiplied by the weight matrix to produce net vector. This vector is then operated on by activation function to produce out vectors for that layer.

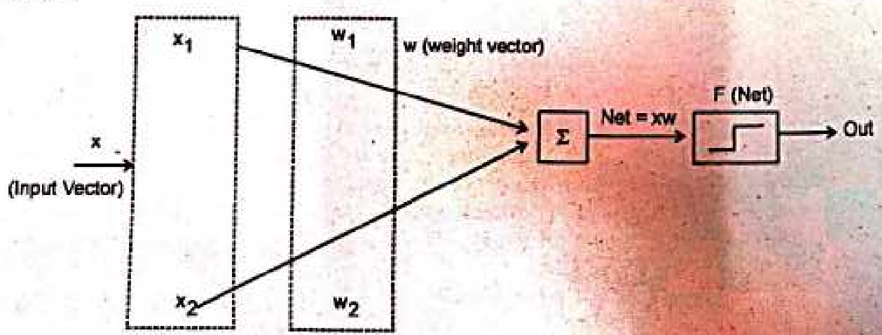


Fig. 4.1 Vector Matrix Multiplier

$$NET = XW$$
$$OUT = F(Net) \text{ Where}$$
X = input row vector
W_i = weight matrix.

In biological neural networks, this operation is performed by large no. of neurons operating simultaneously. The number of operation required for matrix multiplication is proportional to the square of the size of input vector & computational time can be inherently long.

4.3 ELECTRO-OPTICAL MATRIX MULTIPLIERS

Q.4 Explain Electro optical Matrix Multiplier. **(KUK, May 2010)**

Q. 5 Briefly explain Electro-optical matrix multipliers with the help of an example. **(KUK, May 2010)**

Ans. These nets provide a means for performing matrix multiplication in parallel. The network speed is limited only by the available electro-optical components; in this case the computational time is potentially in the nanosecond range. The figure 4.2 shows the electro-optical vector matrix multiplier.

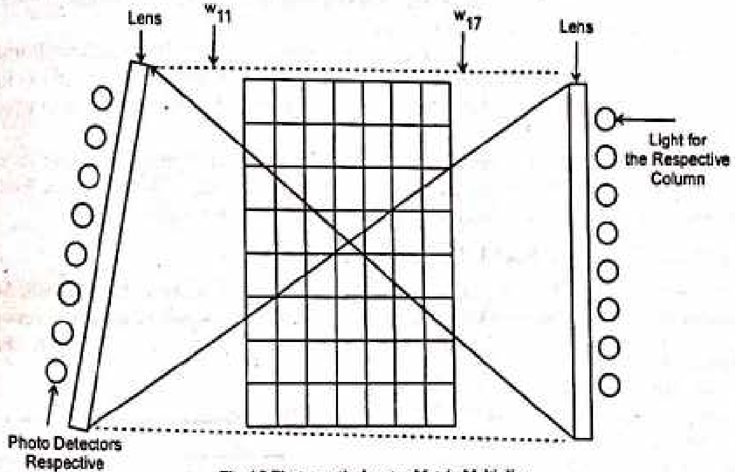


Fig. 4.2 Electro-optical vector Matrix Multiplier

The above shown system is capable of multiplying ar 8-element input vector by 8×7 matrix, which produces seven-element NET vectors. There is a column of light sources that passes its rays through a lens, such that each light-illuminates a single row of weight shield. The weight shield is a photographic film in which the transmittance of each square is propartional to the weight. There is an another lens, which focuses the light from each column of the shield to a conespouliy photo detector. The NET is calculated by,

$$NET = \Sigma\, wik\, x_i$$

where

NET_k = NET output of neuron k.
wik = weight from neuron i to neuron k.
x_i = input vector component i.

The output of each photo detector will represent the dot product between the input vector and a column of the weight matrix. The set of outputs is a vector equal to the product of the input vector with weight matrix. Hence, matrix multiplication is done in parallel. The speed is independent of the size of the array. This makes the network to be scaled up without increasing the time required for computation. The weights may be made variable, to use in adaptive system. For the weights, instead of photographic film, a liquid crystal light value may be used. This permits the weights to get adjusted electronially. This electro-optical matrix multipliers can be used Hopfield net and bidirectional associative memory.

4.4 HOPFIELD NET USING ELECTRO-OPTICAL MATRIX MULTIPLIERS –

If the photo detector outputs of the networks are fed back to drive the corresponding light inputs, an electro-optical hopfield net is produced.

- Threshold activation function is needed for this purpose. To satisfy the stability requirements, the weight array must be symmetrical with transmittance set to zero for squares on main diagonal.
- Electro optical BAM's : If two of the system feeding back to the input of first, an electro optical BAM is produced. To ensure stability, the second weight must be the transpose of first.
- Kosko described a compact system in which only a single mask and optical system is required. Here, each photo detector and light source is replaced by a photo detector light source pair.
- The operation is similar to that of simple optical multiplier except that output from each photo detector deceives its adjacent light source.
- The light from each light source on the right passes through the cylindrical lens illuminating the corresponding row of weight mask on the left, each photo detector receives all the light from a column of weight mask and its electronics provides the threshold function to produce NET output.
- On the right, each photo detector responds to the light from an entire row and its electronics perfoms the threshold function and drive the adjacent light source. In this way a feedback loop is closed, coupling light sources, photo detectors and optical system.

4.5 HOLOGRAPHIC CORRELATORS

Q.6 Write short note on Holographic correlator. (KUK, June 2007- 2008, May 2010)

Q.7 Explain with a neat diagram how holographic correlator are used for image recognition. (KUK, June 2010)

Ans. Holographic Correlator – In this case, the reference images are stored in a thin hologram and retrieve them in a coherently illuminated, feedback loop. A noisy or incomplete input image is applied to the system and can simulataneously be correlated optically with all of the stored reference images. These correlations can be threshold and are feedback to the input, where the strongest correlation reinforces around the loop repeatedly, which approaches the stored images more closely on each pass, up to the system correlator can be used for image recognition. A generalized optical image recognition system is shown in fig.

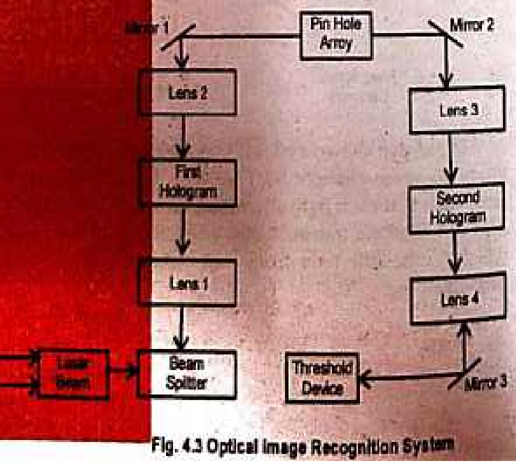


Fig. 4.3 Optical Image Recognition System

The input to the system is an image from a laser beam. This passes through a beam splitter, which passes it to the threshold device. The image is reflected then gets reflects from the threshold device, passes back to the beam splitter, then goes to lens 1, which makes it falls on the first hologram.

The first hologram contains several stored images. The images then gets correlated with each of them, the produces patterns of light. The brightness of the patterns varies with the degree of correlation. The projected image from lens and and mirror 1 passes through pin hole array, at which they are spatially reparated.

From this array light, patterns goes mirror 2, through lens 3 and then applied to second hologram. Lens 4 and mirror 3, then produce superposition of the multiple correlated images onto the back side of the threshold device.

Operation of Threshold Device – Its front surface reflects, most strongly that pattern which is brightest on its on its rear surface. A set of four correlations of each of the four stored images with the input images projected on its rear surface. The stored image which is similar to the input image possess highest correlation. This reflected image then again pass through the beam splitter and re-enters the loop for further enhancement. The system gets converged on the stored pattern most like the input patterm.

This is operation of the holographic optical image recognition system. Here also Hopfield net can be formed using this hologram correlator.

Optical neural network is advantages in terms of speed and interconnect density. They can construct virtually any network architecture.

4.6 VOLUME HOLOGRAMS

Certain crystal will bend and incident light beam,; the amount of bending can be modified by a leaser. If neurons are designed to both receive and transmit light the, photo fractive crystals can be used to interconnect large networks. The potential density of such interconnective systems and estimate that from 108 to 1010 interconnections, per cubic centimeter are possible. The amount 4 direction in which a light beam is bent by a photopractive crystals are determined by interval holographic gratings formed by a neigh intensity laser beam.

The crystal's local index of refraction is a function of its local charge density. The laser redistribute the charge by dislocating electrons thus forming regions of varfing refracting power. If a light ray reconnecting a pair of neurons impinges on the crystal at an appropriate point, it will be bent by the proper angle to direct it to the destination neurons, further more the strength of each grating can be controlled as it is written by laser beam, than by varying the percentage of the incident beam that is refracted. This effectively changes the weight of interconnections, allowing the system weights to be modified by a learning algorithm.

4.7 AN OPTICAL HOPFIELD NET USING HOLOGRAMS

Q.8 Write a technical note an Optical Hopfield net using volume Holograms.(KUK, May 2011)

Ans. An all Optical recurrent neural network using volume holograms has been reported by Stoll and Lee. It operates as an implementation of the Hopfield Net, Seeking a minimum on an optically generated energy surface. When a noisy or incomplete input pattern is applied the system converges to the started image that is most similar, there by functioing as an optical associative memory. A resonant loop encloses the optical neurons aray, the optical interconnect matrix and the associated optical component. Images pass around this feed back loop in the directions indicated by the arrows, being amplified in the process. There is a close analogy here to the operation of the Hopfield network. The optical neuron array sums the input and the feed back signals and then applies the sigmoid activation function. the optical interconnect matrix performs the vector matrix multiplication. when an input vector is applied at the right, it passes through beam splitter to Lens.

L1, where it enters the optical interconnect matrix. A portion of the output light also passes through BSI and constitutes the system output.

4.8 THE COGNITRONS, THEIR STRUCTURE AND TRAINING –

Q.9 Explain the structure and training procedure of the cognitrons. **(KUK, June 2008, May 2011)**

Q.10 Explain the cognitrons. **(KUK, May 2010 (R))**

Q. 11 Write a short note on cognitrons. **(KUK, June 2010)**

Ans. The congnitrons was developed by Fukushima in 1975. it is a hypothetical mathematical model of the human perception system.

Structure - The cognitron is made up of layers of neurons which are connected by rephrases. There exist presynaptic and post synaptic neurons in the next layer. Also, there exist two types of cells, excitatory cells and inhibitory cells. The excitatory cells make the post-synaptic cell to fire and the purpose of inhibitory cells is to reduce the firing of post-synaptic cell. The firing of neuron depends on the weighted sums of its excitatory and inhibitory inputs. The pre-synaptic and post synaptic neurons are shown in fig. 4.4

Also each neuron connects only to neurons in the nearby area, called its connections region. This is

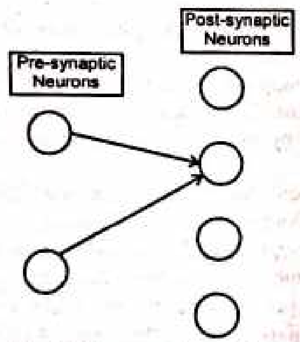


Fig. 4.4 Presynaptic and Post Synaptic Neurons

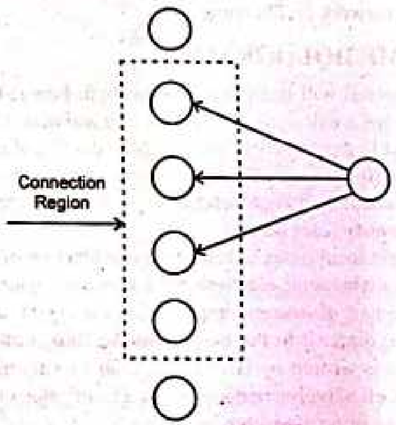


Fig. 4.5 Connection Region of a Neurons

shown in fig. 4.4. In cognition, neurons are arrayed in Layers, with the connections from one layer going to the next layer.

Training – The training employed in cognitron net is unsupesvised training. For a given training set of input patterns, the network self-organizes by adjusting its synaptic strengths. Give it is an unsupervised method of training, there are no predetermined output patterns that represent the desired response for cognition net. In a given region of a layer only the neuron are already well trained, are shown by the strength of their firing. Here, competition among nearby cells is adopted. If the connection regions of nearby cells overlap, then there is a possibility of group of cells to have similar response patterns. Thus to avoid this overlapping, competitions is used.

This is shown in fig 4.6

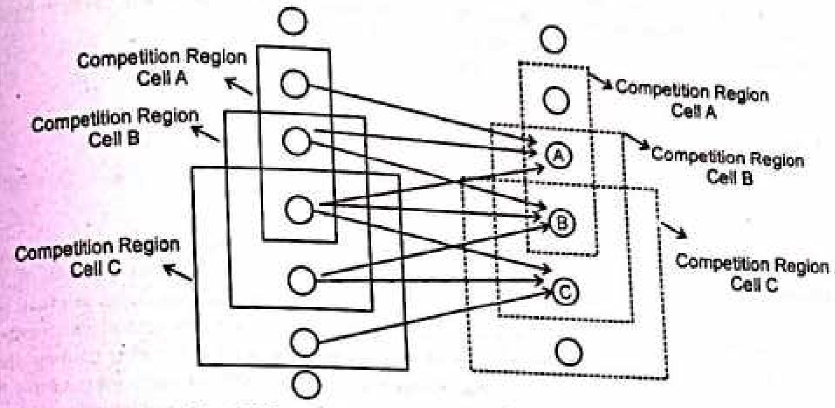


Fig. 4.6 Connection and Competitive Regions

Excitatory Neuron – The output of excitatory cognitions neuron is determined by the ratio of excitatory inputs to inhibitory inputs. The total excitatory input to a neuron E is simply the weighted sum of the inputs from the excitators neurons in previous layer. Also, the total invibitory input to a neuron I, is the weighted sum of the inputs from the inhibitory neurons.

It can be given as,

$$E = \sum w_i x_i$$
$$I = \sum v_j y_j$$

Where

W_i - The weight of the i th excitatory synapse.

x_i - The output of the ith excitatory neuron.

V_j - The weight of the jth inhibitory synapse.

y_j The output of the jth inhibitory neuron.

Here the weights take only positive values. The output of the neuron is then calculated by,

$$\text{Net} = \text{Net input} = \left[\frac{(1+E)}{(1+I)}\right] - 1$$

$$\text{Out} = \begin{Bmatrix} \text{net, for net} \geq 0 \\ 0, \text{ for net} < 0 \end{Bmatrix}$$

Taking only positive value from the net, we get,

$$\text{Output} = \left(\frac{E-1}{1+1}\right)$$

$$\text{Output} = E - 1, \text{ for } I <<< 1.$$

In the cognitron, Large excitatory and inhibitory inputs results in using the below formula.

$$\text{Output} = \left(\frac{E}{I}\right) - 1 \text{ is } E >>> I \text{ and } I >> 1$$

Here, the output is determined by the ratio of the excitatory inputs to inhibitory inputs, rather than taking the difference between the two. Hence, the value of output is limited, provided that both types of input increase at the same rate P. As a result, E and I can be expressed as follows :

$$E = aP$$
$$I = bP$$

Where a, b = constants

using some transformations,

$$\text{output} = \left[\frac{(a-b)}{2b}\right]\left[1 + \tanh\left\{\log\frac{(ab)}{2}\right\}\right]$$

From this relationship, the cognition neuron closely emulates the responses of the biological neuron.

Inhibitory neuron – In cognitron a layer consist of both excitatory and in Inhibitory cells. In fig. 4.7, it is found that a layer 2 neuron has a connection region over which it has synaptic connections to a set of layer 1 neuron gives output signals. Similarly in layer 1, there is an inhititory neuron with the same connection region. The weights coming into inhibitory cells are not modified during training, their weights into any inhibitory neuron is equal to one. Hence the output of the inhibitory cell is simply the weighted sum of its inputs, which in this case is the arithmetic mean of the excitatory output to which it connects. Hence

$$\text{Inhibit} = \sum u_i \text{ output}_i$$

where $\sum u_i = 1$

and u_i = inhibitory weight i

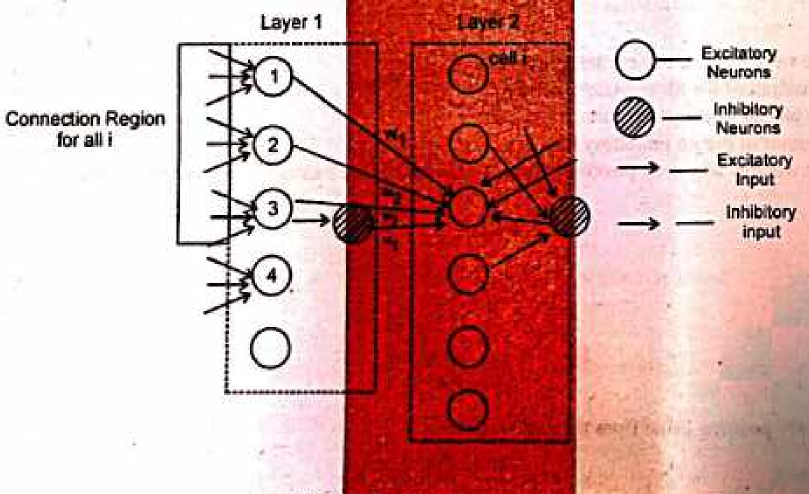


Fig. 4.7 Cognitron Layers

Training Problems – The weights associated with an excitatory neuron are adjusted only when it is firing more strongly than any of the neighboring cells in the competitive region. The change in one of its weights in the case is calculated as,

$$\delta w_i = b\, u_j\, x_j$$

where,

u_j - The inhibitory weight coming from neuron j in layer 1 to the inhibitory neuron i.

x_j - The output of neuron j in layer 1.

wi - Excitatory weight i.

b - Learning rate coefficient.

The chase in the inhibitory weight into neuron i in layer 2 is proportional to the ratio of the weighted sum of excitatoy inputs to twice the inhibitory input. This is calculated using the following formula :

$$\delta V_i = \left(\sum_j w_j x_j\right) / (2 \times \text{Inhibit}_i)$$

When no neuron sizes in the competition region, other weight adjustment formula is used.

$$\delta w_i = b\, u_i\, x_i$$
$$\delta V_i = b \text{ Inhibit } i$$

Where b is a positive training co-efficient more than 1. This adjustment ensures that a cell with a large response cause for excitatory synapses that it drives to increase more than the inhibitory synapses. The training adjusts the weights of each layer 2 cell so that they jointly constitute a template for confirming patterns that were present frequently during training process.

Cognitron being a self organized network failed to recognize position or rotation distarted characters. This failure was overcome in the neocognitron network.

4.9 NEOCOGNITRON, THEIR STRUCTURE AND TRAINING –

Q.12 Write short note on training of Neocognitrons. **(KUK, June 2008)**

Q.13 Explain structure and training of Neocognitron. **(KUK, may 2009)**

Q. 14 Explain application of Neocognitrons for pattern recognition. **(KUK, May 2011)**

Ans. Necognitron- The necognitron is powerful in its ability to recognize patterns despite translation, rotation, distortion and changes the scale. The recognitron is modeled towards the human visual system. It can accept two dimensional patterns like those imaged into the retina and process then in successive layers as that of human visual cortex. It is a hierarchical net in which there are many layers with sparse and localized pattern of connectivity between layers. The neocognitron is based on supervised learning. This net is designed so that it is insensitive to the variations in the position, style etc. of the patterns to considered.

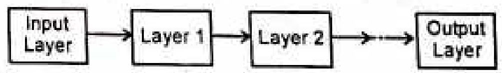


Fig. 4.8 Neocongnitron Layer structure

Structure - The recognition consists of several layers of units as shown in fig 4.8

Each layer has units arranged in number of square arrays. Only very limited number of signals are transmitted from one unit to the other. The layers are arranged in pairs that is there is an S- layer followed by a C- layer as shown in Fig 4.9.

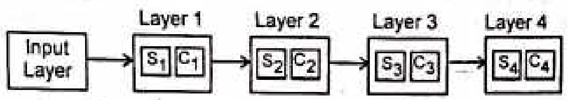


Fig. 4.9 Neocognitron layers

S- Cells : Simple Cells – The S-arrays are trained to respond to a particular pattern of group of patterns. All cells in a simple cell plane respond to the same pattern. Each simple cell is sensitive to a restricted area by the input pattern which is called its receptive range. The receptive ranges of the entire input pattern for that layer.

C- Cells : Complex Cells – The C- cells'combines the outputs from S- cells and simultaneously thin out the number of units in array. complex cells serve to make the system less sensitive to the position of patterns in the input field. The complex cells receive outputs from a set of simplex cells. The simple cells cover a range called as receptive range. Each layer of complex cells responds to a larger range of the input pattern than that in the preceding layer.

The units in each layer are arranged in several square cells. The size of the cells may be given as 19× 19, 7× 7, 16× 16, 3× 3, 5× 5, 1× 1 etc. depends upon the pattern used.

Algorithm Calculations – The s- type cell passes excitatory signals are received from the units in the previous layer and passess inhibitory signals obtained within same layer

$$\vartheta = \sqrt{\Sigma\Sigma t_i c_i^2}$$

Where ti - Fixed weight.
Ci - Output from C unit.
The S- unit has its scaled input as,

$$x = \frac{1+e}{1+VW_0} - 1$$

Where $e = \sum_i c_i w_i$ wi - weights adjusted from c to s.
wo - weights adjusted between V and S.
e - Excitatory input from C units.
The activation of output signal is,

$$S = \begin{cases} x, \text{if } x \geq 0 \\ 0, \text{if } x < 0 \end{cases}$$

The net input of C- layer is,

$$C_{in} = \sum_i s_i x_i$$

Where S_i - Output from S - unit.
xi - Fixed weight from S - uint to C - Unit.
The output is

$$C = \begin{bmatrix} \frac{C_{in}}{a+C_{in}}, & \text{if } C_{in} > 0, \\ 0, & \text{otherwise} \end{bmatrix}$$

'a' is a parameter that depends on the level of the network performance.

Training – There are several different methods for training the weights on neocongnitron. Hence, we will discuss the unsupervised learning algorithm.

Unsupervised Learning – In this method, training proceeds as it for many network. First a input pattern is presented to the network at the output layers data propagated through the network. The weights are allowed to make incremental adjustments according to specified algorithm, After weight updates have occurred, a new pattern is presented to network at the input layers the process is repeated with all patterns in training set untill the network is classifying input vectors properly. In neocongnitron, sharing of weights or a given plane means that only a single cell on each plane needs to participate in the learning process. Once its weights have been updated, a copy of new weight vector can be distributed to other cells on same plane.

--- Suppose S- planes on a given layer are being stacked vertically on top of one another. Now, there will be many overlapping columns number perpendicular to this stack. These columns define groups in S-cell where all members in a group have receptive fields in the same location of input layer. Now, we can apply a new input patterns examine the response of S-cells in each column to ensure that each S-cell provides a distinct response, we initialize the weights to small, positive random values.

--- The weights on inhibitory connections to be initialized to zero. Now, we find the plane position of S-cell whose response is strongest in each column. Similary, in thus manner we will locate the S-cell on each plane whose response is strongest subject to condition that each of these cell is in a different column. Those S-cell become the respresentation of all cell on their respective planes. Similarly, the strongest responding V_t— Cell is chosen representative are chossen, the weight updates are made according to following equation.

$$\Delta ae\{ke-1, u, Ke\}\ q_e c_{e-1}(v) u_{c-1}[k_{e-1}, n+v]$$

$$\Delta b_e[K\hat{e}] = q_e' c_{e-1}(\hat{n})$$

Where q_e-learning rate parameter

---The largest increases in weights occur on those connections that have largest input signal, $Vc^{e-1}[K_{e-1, n+v}]$ Because S-cell whose weight are being modified was the one with largest output. The weights can only increase, there is no upper bond on weight value.

---Once the cells on a given plane, begin to respond to a certain feature they tend to respond less to other features.

4.10 FUZZY LOGIC

Q.15 Explain Fuzziness of Fuzzy sets. (KUK, May 2009)

Q. 16 Write a short note on "Fuzziness of Fuzzy Sets". (KUK May 2010)

Ans. Fuzzy logic is a super set of conventional (or Boolean) logic and contains similarities and differences with Boolean logic. Fuzzy logic is similar to Boolean logic, in that Boolean logic results are returned by Fuzzy logic operations when all fuzzy memberships are restricted to 0 and 1. Fuzzy logic differs from Boolean logic in that it is permissive of natural language queries.

If a proposition is true, it has a truth value of 1 and if it is false then its value is 0. proposition can also be combined to generate other propositions by means of logical operators. There are certain statements the truth values of which lies between 0 & 1. For eg. consider the statement "It will rain today." the level of certainty lies between 0 & 1. Fuzzy logic was developed to model such situation. Fuzzy logic was developed to model or deal with propositions that one true to a certain degree.

Fuzzy values need not be add up to 1.0 like probability values. For eg. the truth value of propositions that a cup of tea hot is 0.8 and the truth values of the statement the cup of coffee is cold is 0.5. FL incorporates a simple, rule-based If X AND Y then Z approach to solving a control problem rather than attempting to model a system mathematically.

Fuzzy Set- A fuzzy set is a set whose elements have degrees of membership. That is, a member of a set can be full member (100% membership status) or a partial member (eg. less than 100% membership and greater than 0% membership). To supply understand fuzzy sets, one must first understand the traditional sets.

A traditional or crisp set can formally be defined as the following.

- A subset U of a set S is a mapping from the elements of S to the elements of the set {0,1}. This is represented by the notation. V : S---> {0,1}
- The mapping is represented by one ordered pair for each element S, where the first element is from the set S and the second element is from the set {0,1}. The value zero represents non-membership while the value one represent membership.

  Again a fuzzy set is a set elements have degree of membership. These can formally be defined as the following:
- A fuzzy subset F of a set S can be defined as a set of ordered pairs.

The first element of the ordered pair is from the set S, and the second element from the ordered pair is from the interval {0,1}.

- The value zero is used to represent non-membership; the value one is used to represent complete membership and the values in between are used to represent degrees of membership.

Fuzzy set operations – The usual operations we can perform on ordinary sets are union in which we take all the elements in the fuse set and the other. Second is intersection in which we take elements which are common to both sets.

— In case of fuzzy sets, taking a Union is finding the degree of membership that an elements should have in new fuzzy set.

— If a, b, c, d are such that their degrees of membership in fuzzy set A are 0.9, 0.6, 0.8, 0.3 m them fuzzy set A is given by fit vector (0.9, 0.6, 0.8, 0.3). The values in the fit vector are called as fit values.

Union of fuzzy set – To find the Union of fuzzy set, we need to look at the degree of membership of the sets. The elements with higher degree of membership are listed in the Union while others are left, eg. consider two sets

A = {0.9, 0.4, 0.3, 0} & B = {0.2, 0.6, 0.3, 6.1}

then A [CUP] B = {0.9, 0.6, 0.8, 0.8}

Where Cup = Union of A & B.

Intersection and Complement of Fuzzy sets – To find the intersection of two suzzy sets, find the minimum or smaller values of its degree of membership individually in two sets forming no intersection.

Let

A = {0.9, 0.4, 0.5, 0} B = {0.7, 0.6, 0.7, 0.8}

A [Cap] B = (0.7, 0.4, 0.5, 0)

When cap = intersection.

The complement of a fuzzy set is obtained as follows. If the degree of membership is 0.8 for a member then that member is not that set to a degree of 1.0-0.8 = 0.2. So, the degree of complement of an element in the set is found out w.r.t. 1.0

Let = A (0.9, 0.4, 0.5, 0)

Then Let A' denote the complement of A then

A' = (0.1, 0.6, 0.5, 1)

and Let B = (0.7, 0.6, 0.3, 0.8)

Then B' = (0.3, 0.4, 0.7, 0.2)

Now A' [Cup] B' = (0.3, 0.6, 0.7, 0.8)

Fuzzi fication – Establishes the fact base of the Fuzzy system first, it identifies the input and output of the system. Fuzzification then defines appropriate IF THEN rules and uses raw data to derive a membership function. At this point, one is ready to apply fuzzy logic to the system.

Consider an air conditioning system, that samples the current temperature and moisture levels to determine the optional circulation level. In this case, the input consists of the current temperature and moisture level. The fuzzy system outputs the optional air circulation level : "none", "Low" or "high".

The following fuzzy rules are used :

(1) If the room is hot, circulate the air a lot.

(2) If the room is cool, do not circulate the air.

(3) If the room is cool and moist, circulate the air slightly.

Applications of fuzzy sets : Fuzzy logic is well suited for system that repair the ability to handle vague data and/or model imprecise reasoning procedures. Many commercial applications of fuzzy logic relate to "process control" which refers to the management of a mechanical or environmental process. Examples of fuzzy logic technology include :

(1) **Air conditioning –** Gradually slows down the cooling system as the room temperature approaches the desired setting.

(2) **Cruise Control –** determines ambient acceleration or declaration and controls the countering application of gas and brake.

(3) **Ship boilers –** monitors the temperature, pressure and chemical content to ensure stability.

(4) **Video cameras –** identifies when the subject of a video shot is moving and when motion is caused by the camera man's vibrations.

(5) **Washing machines –** Optimizes the wash cycle by examining the load size, fabric mix, and quality of detergent.

4.11 GENETIC ALGORITHMS, ELEMENT OF GENETIC ALGORITHM, WORKING OF GENETIC ALGORITHMS

Q.17 Give the elements of Genetic algorithm. Explain working of genetic algorithms evolving neural networks. (KUK, June 2007)

Q.18 Write short note on genetic algorithm. (KUK, June 2008, 2010)

Q. 19 Explain Genetic algorithm. (KUK, May 2009)

Q. 20 Explain main operators of Genetic algorithms. (KUK, May 2011)

Ans. Genetic Algorithms – Gentic Algorithms are the algorithms that dictate how population of organisms should be formed, evaluated & modified.

For eg. There is a genetic algorithm that determines how to select organisms for sexual reproduction & another that will determine which organism will be deleted from population.

The problems that may be solved by genetic algorithms very form a varity of data mining teachniques such as neural nets to optimization of negotiation strategies for all right.

— The trick is to determine how to convert proposed solution to a real world problem into simulated genetic material on a computer.

Elements of Genetic Algorithm – There is no rigorous definition of genetic algorithm accepted by all in the evolutionary. computation community. However, it is said that most Genetic Algorithms have following elements in common Population of chromosomes, selection according to fitness, crossover to produce now offsring & random mulation of new offsring.

— The chromosomes in a GA population typically take the form of bit strings. Each Louis in the chromosome has two possible alleles : 0 & 1.

— Each chromosomes, can be thought of as a point in the search space of candidate solutions. The GA processes population of chromosomes successively replacing one such population with another.

—The GA most often require a fitness function that assigns a score to each chromosome in current population. The fitness depends on how will the chromosome solves the problem at hand.

(a) **Example of Fitness Function :** Suppose one wants to maximize a real valued one dimensional function:

$$f(y) - y + |\text{Sin}\ \{32y\}|$$

Here the candidate solutions are values of y which can be enclosed as bit strings the represantating real no. The fitness calculation translates a given bit string X into real no. y_s then evaluates function at that value. The fitness of a string is a function value at that point.

(b) **GA Operators –** The simplest form of genetic algorithm involves 3 operators : selection, cross over smulation.

1. **Selection –** This operator select chromosomes in the poulation for reproduction.
   — The fitter the chromsomes, the more times it is likely to be selected to reproduce.
2. **Crossover –** This operator randomly choses a locus & exchanges the subsequences before & after that locus between two chromosomes to create two offspring.

For eg. the strings 10000100 & 11111111 called be crossed over after third locus in each to produce two offspring 10011111 & 11100100.

3. **Mutation** – This operator randomly flips some of the bits in a chromosomes. For eg. the string 0000100 might be mutated in its second position to yield 01000100.

— Mulation can usually occur at each bit in the string with probabillity, usually very small.

A simple Genetic Algorithm – Given a clearly defined problem to be solved and a bit string representation of candidate solutions, a simple GA works as follows :

1. Start with a randomly generated population of n-bit chromosomes.
2. Calculate the fitness f(n) of each chromosome x in the population.
3. Repeat following steps until n- offsprings have been created.

(a) Select a pair of parent chromosomes from current population probability of solution being an increasing function, solution is done with replacement.

With probability Pe, crossover the pair at a randomly chosen point to form two offspring. If no crossover takes space form two offspring that are exact copies of their respective parents.

Mutate the two off spring at each locus with probability P_m and place resulting chromosomes in new population.

4. Replace current population with new population.
5. Go to step 2.

Each iteration of this process is called a generation. At the each iteration there are one or more chromosomes that fit highly in the popilation.

—The procedure described above in the basis for most applications of GA's. The success of the algorithm often depends upon details such as size population & probabilities of crossover & mutation.

—Consider an example of a simple GA. Supposse that string length in 8, that f(x) is equal to the number of ones in the bit string x, that n, is 4, Pc = 0.7 & that Pm = 0.001.

The initial population might look like this.

Chromosome Label	Chromosome's String	Fitness
A	00000110	2
B	11101110	6
C	00100000	1
D	00110100	3

A common selection method in GA's is fitness proportionate selection in which the number of times an individual is expected to reproduce is equal to the fitness divided by average of fitnesses in population.

— A simple method for implementing fitness proportionate selection is "routette wheel sampling" which is equivalent to giving each individual a slice of circular routette wheel equal in area to individual fitness.

The roulette wheel is spun, the ball comes to rest on one wedge shapped slice, that corresponding individual is selected.

In the above example, roulette wheel is spun, the ball comes to rest on one wedge shaped slice, that corresponding individual is selected.

In the above example, roulette wheel is spun 4 times, the first 2 spins might choosen B & D are parents's the next two spin might choose B & C to be parents.

— Once the pair of parents is selected, they cross to produce two offsprings. If they not then springs exact copier of parent.

—Suppose B & D cross over after the first bit in string to produce two off springs E = 10110100 & F = 01101110 & parents B & C do not cross over instead forming off spring that are exact copies of B & C. Next each off spring is subject to mutation at each locus with probability Pm. Suppose off spring E is

mutates at sixth locus to form E' = 10110000 & Off spring B is mutated at first locus to form B' = 01101110. The new population will be following :

Chromosome Label	chromosone' String	Fitness
E'	10110000	3
F	01101110	5
C	00100000	1
B'	01101110	5

Thus, in the new population the best string is last but the average fitness raise from 12/4 to 14/4.

Working of Genetic Algorithm – A genetic algorithm beings by creating an initial population. This population consists of chromosomes that are given a random collection of genes. The steps involved genetic algorithm are:

1. Create an initial population of chromosomes.
2. Evaluate the fitness of each chromosomes that makes the population.
3. Based on this fitness, select chromosome that will make.
4. Cross over or mate the selected chromosomes to produce the off spring.
5. Randomly mutate some of the genes of chromosomes.
6. Repeat steps 3 through 5 until new population is created.
7. Algorithm ends when best solution has not changed for a preset number of generations.

Initial Population – In a genetic algorithm population is comprised of organisms. Each of these organisms is usually comprised one complete solution to defined problem.

--- The genetic algorithm must create the initial population which is comprised of genes and these genes are initialized to random values based on boundaries defined.

--- Each Chromosome in the initial population must be evaluated which is done by evaluating the fitness of each chromosome and represents the quality of solution.

--- The fitness is determined by a function and is specific to the defined problem genetic algorithm is designed to solve.

2. **Suitability and privilege to mutate** – The purpose of mating is to create a new improved population. Suitability to mate refers to those chromosomes that are qualified to mate or those who have privilege to mate.

--- Determining the specific chromosomes that will mate is based upon individual chromosome fitness.

— The chromosomes are selected from old population, mate and produce new chromosomes. The new children are joined with existing population.

3. **Mating** – Mating works by taking two parents and taking a splice from their gene sequence. This splice effictively divide the chromosome into 3 gene sequences.

— The children will be built based on genes from each of these 3 sections.

—.The process of mating simply jumbles the genes from these two parents into a new off spring chromosome which allows new chromosomes to take traits from each parent.

--- This method can also lead to the problem, no genetic material being introduced and for that purpose, mutation is used.

4. **Mutation** – It allows new genetic patterns to be introduces. That were not already contained in population. Mutation can be thought of as natural experiments to introduce some what random sequence of genes into a chromosome.

— It is complete by unknown if this mutation will produce a desirable or undesirable attribute.

---- Natural selection will determine the fate of mutated chromosome. If the fitness of mutated chromosome is higher than general population, It is likely to survive and will be allowed to mate with other chromosomes.

— If the genetic mutation produces an undesired feature, natural selection will ensure that the chromosome doesn't live to mate.

— Mutation level is generally expressed as percent. If it is too high, we will be performing nothing more than a random search.

4.12 EVOLVING NEURAL NETWORKS

Neural networks are biologically motivated opproaches for machine learning inspired by ideas from neuroscience. Some efforts have been made to evolve aspects of neural networks by use of genetic algorithms.

— In feed forward a neural network is a collection of connected activitable units in which connections are weighted usually real-valued weights presented with an activation pattern on its input units. Activation spreads in forward direction from input units through one or more layers of hidden units to the output units over weighted connectious.

— There are many ways to apply GA's to neural networks some aspects that can be evolved are weights in the networks, network architecture and learning rule used by the network.

(a) Evolving weights in a fixed networks – Montana and Davis were using the GA instead of bach propagation as a way of finding a good set of weights for a fixed set of connections.

— There were several problems associated with back proposation like stuck in local optima, Unavailability of a teacher etc.

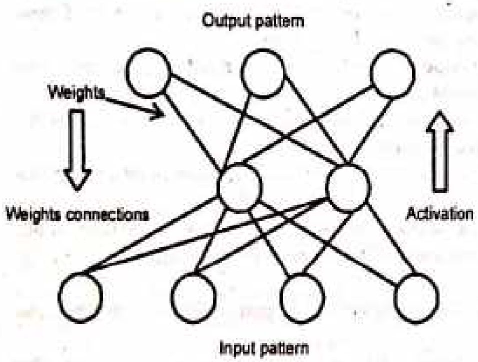


Fig. 4.10 A simple feed forward Network

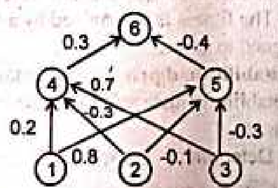


Fig. 4.11 Chromosome : (0.3, -0.4, 0.2, 0.8, -0.3, -0.1, 0.7, -0.3

— Montana and Davis were interested in using neural network to classify underwater sonic "lofargrams" into one of the two classes interesting and non-interesting. The overall goal was to detect interesting signals in the midst of wide variety of noise which exist in the ocean.

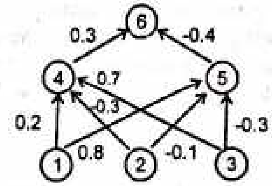
— Each network had 4 input units, 2 hidden units and one output unit which were fully inter connected. In total, there were IOP weighted connections between units. In addition that were 18 connections between non-input units and threshold units. Thus, a total of 126 weights.

— The network was used as follows : Each Chromosome was a list of 126 weights. The weights were read off the network in a fixed order and placed in a list. Each gene in the chromosome is a real no. rather than a bit.

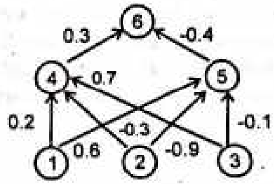
— To calculate the fitness of chromosome weights in the chromosome were assigned to links in corresponding network, the network was run on training set and sum of squares of errors was returned.

— An initial population of 50 weights vector was selected with each weight between 0 and 1. The mutation and cross over operators were used for, comparison of GA with back propagation.

— Mutation operator selects a n- non input units and for each unit with an incoming link adds a value between 0 and +10 to weights on the link.

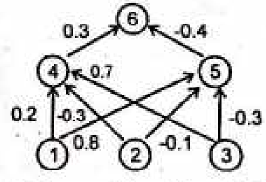


Before restation (a)
Fig. 4.12 Weights coming to unit 5 are mutated

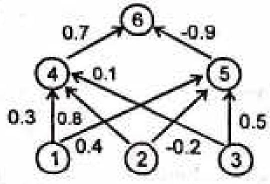


After restations (b)
Fig. 4.12 Weights coming to unit 5 are mutated

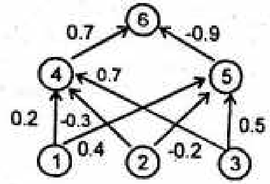
— The cross over operator selects two parent weight vector and for each non-input unit in offspring vector, selects one of parents and copies the weight on the incoming links to that offspring.



Before restations Fig. 4.12 (c)



After restations Fig. 4.12 (d)



After restations Fig. 4.12 (e)

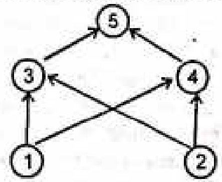
(b) Evolving network architecture– In most network applications, the architecture of network- the number of units and interconnections is decided ahead of time by programmer by network often aided by some heuristic and trial errors.

— Neural researches know too well that how a particular network chosen can determine the success or failure of application so they like to optimize the procedure of designed the architecture for an application.

— GA's have also made efforts along these lines which fall into one of two categories—direct encoding & grammer encoding.

(1) Direct Encoding – In this method, network architecture is directly encoded into a GA cromosome. Consider the example : from unit :

to unit	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	L	L	0	0	0
4	1	1	0	0	0
5	0	0	L	L	0



Chromosome – 0000000000011000010000. In this example, connection topology is represented by a 5× 5 matrix in which each entry encodes the type of connection. 0 means no connection, L means a learnable connection that were specified to be learnable were initialized to small random weights in network. Moller and Todd used a simple fitness proportionate method and mutation. Their crossover operator randomly choose a row index and swapped corresponding rows between parents to create offsprings. The fitness of the chromosome was the sum of squares of eross at last epoch. Low error translated to high fitness.

(2) Gramatical encoding – One of the problems of direct encoding approch is that as size of n/w increases, it becomes extremely difficult to use this approach because the size of required chromosome increases very quickly which leads to problems in both performance and accuracy.

--- The solution pushed by kitano is to encode networks as grammars, the GA evolves the grammars but the fitness is tested only often a development step in which a network develops from the grammar.

--- A grammar is a set of rules that can be applied to produce a set of structure. A simple example grammer :

$S \longrightarrow a\,sb$

$s \longrightarrow E$

Here S is the start symbol and E is null string. To produce a structure, stat with S and go on applying the rules untill desired structure is obtained.

(c) Evolving a learning rule – Chalmers used GA's to evolve a good learning rule for neural networks. He limited his initial study to fully connected feed forward nets with only input units and output units i.e. no hidden units. A learning rule is used during training procedure to evolve weights in a network. A learning rule for a single layer feed forward net uses following local information :

a_i - activation of o/p unit j.

o_j - activation of output unit j.

t_j - training signal j.

w_{ij} - current weight on link from i to j.

Change to make in weight

$\Delta w_{ij} = n(t_j o_j - a_i o_j)$

Chalmers in GA population encoded such functions.

— Chalness made the assumptions that this rule should be a linear function of these variables and all their pair wise products

$\Delta w_{ij} = k_0 (k_1 w_{ij} + k_2 a_i + k_3 a_j + k_4 b_j + k_5 w_{ij}\, ai + k_7 w_{ij}\, aj + k_7 w_{ij}\, b_j + k_8\, a_i a_j + k_9\, a_i t_j + k_{10} a_j t_j)$

where k_5 to k_{10} are constant coefficients.

— One goal of chalmers was to see if the GA could evolves a rule that performs as well as delta rule.

Q. 21 Explain RTOS? **(KUK, May 2009)**

Ans. **RTOS** – RTOS means real time operating system. Timeliness is the single most important aspect of a realtime system. These system respond to a series of external inputs, which arrive in an unpredictable fashion. The real time system process these inputs, take appropriate decisions and also generate output necessary to control the peripherals connected to them. These systems are characterized by having time as a key parameter. Often there are hard deadlines that must be met. If the timing constraints are not met, system failure is said to have occured. It is essential that the time constraints of the system are guaranteed to be met which required the system to be predictable. The design of a real-time system must specify the timing requirements of the system and ensure that the system performance is both timely and correct.

There are three types of real-time systems based upon the time contraints :

(a) Hard real-time system – These types of system have fixed deadlines and the jobs must be completed with in those deadlines. For example, the late response of medical equipment analyzing human body will fail if the response is not made within a certain deadline.

(b) Soft real-time system – These type of system don't have fixed deadlines. In these system missing an occasional deadline is acceptable but repeated late computations can result in system failures. For example : An example of such a system includes airlines reservation system.

(c) Fire real time system – These type of systems are a combination of both hard and soft real time systems. These systems have a shorter soft requirement and a longer hard requirement. For example, A patient ventilator must mechanically ventilate the patient in certain amount of time of a given period. A few second delays is allowed but not beyond a certain time period.

Most realtive systems interface with and control hardware directly. The software for such systems is mostly custom developed. These system often don't have standard peripherals connected with a desktop computer like keyboard, mouse etc. Real-time systems often have a customized version of these devices.

Real time system are abbreviated as RTOS.

There are various services provided by the real time systems.

1. Task management and scheduling
2. Interrupt servicing
3. Hardware interrupts
4. Software Interrupt
5. Communication and synchronization
6. Memory Management

Q. 22 Explain Interprocess communication? **(KUK, May 2009)**

Ans. **Interprocess communication** – A third function of real time operating system is commonly known as the name of interprocess communication (IPC). It collects a large set of programming primitives that the operating system makes available to tasks that need to exchange information with other tasks or synchronize their actions. Some tasks also need to talk to other computers or to peripheral hardware which involves devices such as serialines or a network and special purpose device drivers.

Interprocess communication (IPC) includes thread synchronization and data exchange between threads beyond the process boundries. If thread belong to the same process, they execute in the same address space i.e. they can access global (static) data or help diectly, without the help of the operating system. However, if threads belong to different processes, they cannot access each other address spaces without the help of the operating system. However, if threads belong to diferent processes, they cannot access each other address spaces without the help of the operating system.

There are two fundamentally different approaches in IPC :

--- Processes are residing on the same computer

--- Process are residing on different computers.

The first case is easier to implement because process can share memory either in the user space or in the system space. This is equally true for uniprocessors and multiprocesses.

In the second case, the computers donot share physical memory, they are connected via I/o devices. Therefore the processes residing in different computer can not use memoring as a means for communication.

IPC via messages requires a data structure that is copied from the space of the sender process into a message buffer in system space, then copied again from the buffer in system. Space to the structure in space of the receiving process.

Q. 23 Explain Threads? **(KUK, May 2010)**

Ans. In traditional operating system, each process has an address space and a single thread of control. Infact, that is almost the definition of a process. Nevertheless, there are frequently situation in which it is desirable to have multiple threads of control in the same address space running in quasi-parallel, as though they were separate processes.

The process model is based on two independent concepts: resource grouping and execution. Sometimes it is useful to separate them, this is where threads come in. One way of looking at a process is that it is was to group related resources together. A process has an address space containing program text and data, as well as other resources.

The other concept a process has is a thread of execution, usually shortened to just thread. The thread has a program counter that keeps track of which instruction to execute next. It has registers, which hold its current working variables. It has a stack, which contains the execution history, with one frame for each procedure called but not yet returned from. Although a thread must execute in some process, the thread and its process are different concepts and can be treated separately. processes are scheduled for execution on the CPU.

What threads add to the process model is to allow multiple executions to take place in the same process environment, to a large degree independent of one another. Having multiple threads running in parallel in one process is analogous to having multiple processes running in parallel in one computer. Because threads have some of the properties of processes, they are sometimes called light weight processes. The term multithreading is also used to describe the situation of allowing multiple threads in the same

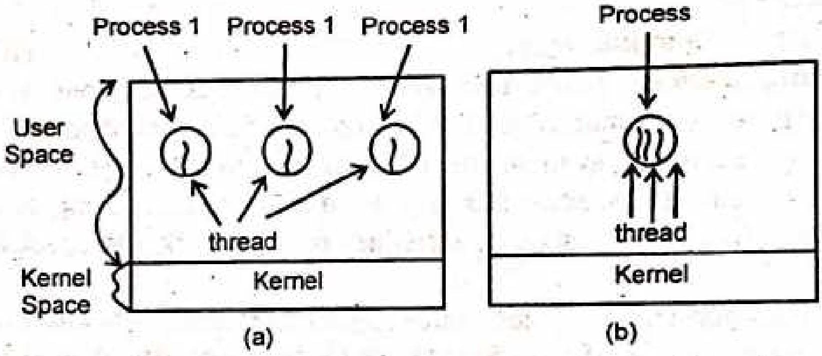


Fig. 4.13 (a) Three processes each with one thread (b) one process with three thread

process.

In fig 4.13 (a) we see three traditional processes. Each process has its own address space and a simple thread of control. In contrast, in fig 4.13 (b) we see a single process with three thread of control. Although in both cases we have three threads. In fig 4.13 (a) each of them operates in a different address space, where as in fig 4.13 (b) all three of them share the same address space.

Different threads in a process are not quite as independent as different processes. All threads have exactly the same address space, which means that they also share the same global variables. Since every thread can access every memory address with in the process, address space, one thread can read, write or even completely wipe out another thread's stack. There is no protection between threads because (1) It is impossible, and (2) It should not be necessary. Unlike different processes, which may be from different usess and which may be hostile to one another, a process is always owned by a single user, who has presumably created multiple threads so that they can cooperate, not fight.

• • •