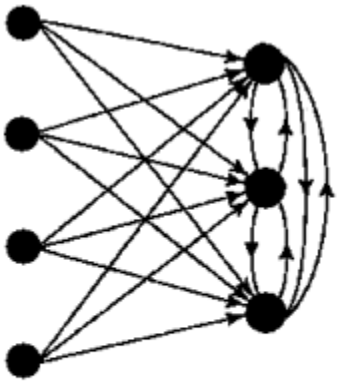
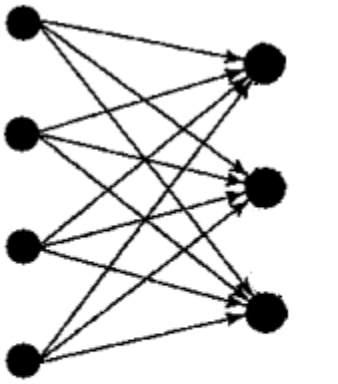
NEURAL NETWORKS

CHAPTER – 6 Competitive Learning Neural Networks

Introduction

Competitive learning neural network consists of an input layer of linear units. The output of each of these units is given to all the units in the second layer (output layer) with adaptive (adjustable) forward weights. The output functions of the units in the second layer are either linear or non linear depending on the task for which the network is to be designed. The output of each unit in the second layer is fed back to itself in a self – excitatory manner and to the other units in the layer in an excitatory or inhibitory manner depending on the task. The weights on the connections in the feedback layer are non-adaptive or fixed. Such a combination of both feed forward and feedback connection layers results in competition among the activations of the units in the output layer and hence such networks are competitive learning neural networks.





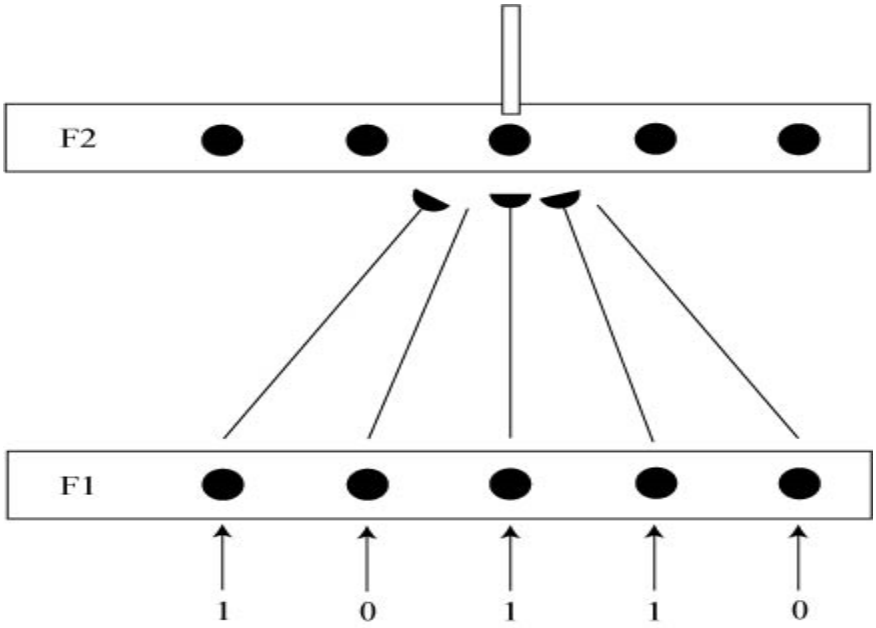
Different choices of the output functions and interconnections in the feedback layer of the network can be used to perform different pattern recognition tasks. If the output functions are linear, then the network performs the task of storing an input pattern temporarily. If the output functions of the units in the feedback layer are made non-linear, then the network can be used for pattern clustering. The objective in pattern clustering is to group the given input patterns in an unsupervised manner and the group for a pattern is indicated by the output unit that has a non zero output at equilibrium. The network is called a pattern clustering network and the feedback later is called competitive layer. The unit that gives the nonzero output at equilibrium is called the winner.

If the output functions of the units in the feedback layer are non linear and the units are connected in such a way that connections to the neighboring units are all made excitatory and to the farther units inhibitory, the network can perform the task of feature mapping. The resulting network is called self – organization network. A self – organization network can be used to obtain mapping of features in the input patterns onto a one- dimensional or a two- dimensional feature space.

Components of a Competitive Learning Network

A competitive learning network consists of an input layer with linear units, a group of instars forming a feed forward portion of the network and a feedback layer with linear or non linear units.

a) *The Input Layer* – The purpose of this layer is to distribute the given external input pattern vector to the feed forward portion of the network. The input vectors may be of varying magnitude even though they contain the same pattern information. The input layer should not feed back ground noise to the feed forward portion of the competitive learning network.
b) *In star* – Each unit in the feedback layer receives inputs from all the input units. A configuration where a unit receives weighted inputs from several units of another layer is called an in star.



c) *Basic competitive learning* – The steady activation value with an external input depends on the angle between the input and weight vectors.
d) *Feedback layer* – In the arrangement of a group of in stars, the category of an input vector can be identified by observing the in star processing unit with maximum response. The maximum response unit can be determined by the system itself if the outputs of the

in star processing units are fed back to each other. If the processing units of a group of instars are connected as an on – centre off – surround feedback network then the feedback layer is called a competitive layer.

Analysis of Pattern Clustering Networks

A competitive learning network with non linear output functions for units in the feedback layer can produce at equilibrium larger activation on a single unit and small activation on other units. This behavior leads to a situation where the input pattern is removed, only one unit in the feedback layer will have non- zero activation. That unit may be designated as the winner for the input pattern. If the feed forward weights are adjusted, each of the units in the feedback layer can be made to win for a group of similar input patterns. The corresponding learning is called competitive learning. The units in the feedback later have non linear output functions.

$$F(x) = x^n , n>1$$

These units are connected among themselves with fixed weights in an on – centre off surround manner. Such networks are called competitive learning networks. Since they are used for clustering or grouping of input patterns, they can also be called pattern clustering networks.

In the pattern clustering task, the pattern classes are formed on unlabelled input data and hence the corresponding learning is unsupervised. In the competitive learning, the weights in the feed forward path are adjusted only after the winner unit in the feedback layer is identified for a given input pattern.

The activation of the i^{th} unit in the feedback layer for an input vector $x =(x_1,x_2,\ldots,x_M)^T$ is given by

$$y_i = \sum_{j=1}^{M} w_{ij} x_j$$

where w_{ij} is the (i, j)th element of the weight matrix W connecting the jth input to the ith unit.

Let i=k be the unit in the feedback layer such that

$$y_k = \max_i(y_i)$$

---

then $w_k^T x \geq w_i^T x$ for all i

If the weight vectors to all the units are normalized, that is $||w_i|| = 1$ for all i, the above result means that the input vector x is closest to the weight vector w_k among all w_i.

$$||x-w_k|| \leq ||x-w_i|| \text{ for all i}$$

If the input vectors are not normalized, then they are normalized in the weight adjustment formula as –

$$\Delta w_{kj} = \eta[(x_j/\sum x_i) - w_{kj}] \text{ only for those j for which } x_j=1$$

This can be called minimal learning.

In the case of binary input vectors, for the winner unit, only the weights which have non zero input would be adjusted that is

$$\Delta w_{kj} = \eta[(x_j/\sum x_i) - w_{kj}] \text{ for } x_j=1$$

$$\Delta w_{kj} = 0 \text{ for } x_j = 0$$

In the minimal learning there is no automatic normalization of weights after each adjustment that is

$$\sum_{j=1}^{M} w_{kj} \neq 1$$

The unit i with an initial weight vector w_i far from any input vector may never win the competition. Since a unit will never learn unless it wins the competition, another method called leaky learning law is proposed. In this case, the weights leading to the units which do not win also are adjusted for each update as follows-

$$\Delta w_{ij} = \eta_w[(x_j/\sum x_m)-w_{ij}] \text{ for all j} \quad \text{If i wins the competition, that is i=k}$$

$$\Delta w_{ij} = \eta_l[(x_j-\sum x_m)-w_{ij}] \text{ for all j} \quad \text{if i loses the competition that is } i \neq k$$

Where η_w and η_l are the learning rate parameters for the winning and losing units respectively ($\eta_w >> \eta_l$). In this case, the weights of the losing units are also slightly moved for each presentation of an input.

Analysis of Feature Mapping Network

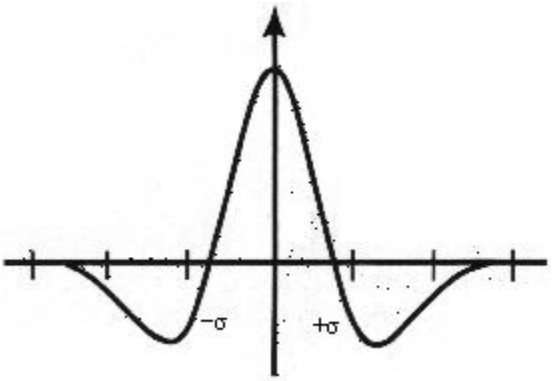
Sometimes, it is difficult to group the input patterns into distinct groups. The patterns may form a continuum in some feature space. It is therefore necessary to have some order in the activation of the unit in the feedback layer in relation to the activations of its neighboring units. This is called feature mapping. The network that achieves this is called feature mapping network. A feature mapping network is also a competitive learning network with non linear output functions for units in the feedback layer.

The inputs to a feature mapping network could be N-dimensional patterns, applied one at a time and the network is to be trained to map the similarities in the set of input patterns. Another input pattern is where inputs are arranged in a 2-D array so that the array represents the input pattern space. At any given time, only a few of the input units may be turned on that is only the corresponding links are activated. The objective is to capture the features in the space of the input patterns and the connections are in the unsupervised learning mode in which the network must work.

Ways of implementing feature mapping process

In one method, the output layer is organized into predefined receptive fields and the unsupervised learning should perform the feature mapping by activating appropriate connections. This can also be viewed as oriental selectivity.

Another method is to modify the feedback connections on the output layer. Instead of connecting them in an on – centre off surround manner, the connections can be made as indicated by a Mexican hat type function.



The function gives the lateral connection weight value for a neighboring unit k at a distance |i-k| from the current unit i.

A third method of implementing the feature mapping network process is to use an architecture of a competitive learning network with on centre off surround type of connections among units, but at each stage the weights are updated not only for the winning unit but also for the units in its

neighborhood. The neighborhood region may be progressively reduced during learning. This is called self organization network with Kohenan's learning.

CHAPTER – 7 Architectures for Complex Pattern Recognition Tasks

Introduction

Functional units are basic structures of neural networks capable of performing simple pattern recognition tasks. Designing architecture for a given task involves developing a suitable structure of the neural network and defining appropriate activation and synaptic dynamics.

Networks are organized at the architectural level along the broad categories of learning (supervised and unsupervised) and along the broad categories of structures (feed forward and feed back).

In supervised learning, the weight adjustment at each step is based on the given input and desired output. The adjustment may be of correlation type, perceptron learning, delta learning, reinforcement learning etc.

In unsupervised learning, the weights of the network are determined either by learning or by computation from the given input patterns.

In feed forward structure, the pattern recall is an application of the computations involved for each unit.

In feedback structure, the pattern recall is involves several cycles of computations, each cycle consisting of computations by all the processing units on the average.

Associative memory

Pattern storage is a pattern recognition task where it is a memory function and the network is expected to store the pattern information for later recall. The patterns to be stored may be of spatial type or spatio- temporal (pattern sequence) type. An artificial neural network behaves like an associative memory in which a pattern is associated with another pattern or with itself.

The pattern information is stored in the weight matrix of a feedback neural network. If the weight matrix stores the given patterns, then the network becomes an **auto associative memory**. If the weight matrix stores the association between a pair of patterns, then the network becomes a **bidirectional associative memory**. This is called hetero association between two patterns. If the weight matrix stores multiple associations among several (>2) patterns, then the network becomes **multidirectional associative memory**. If the weights store the associations between adjacent pairs of patterns in a sequence of patterns, then the network is called a **temporal associative memory.**

Bidirectional Associative Memory

The objective is to store a set of pattern pairs in such a way that any stored pattern pair can be recalled by giving either of the patterns as input. The network is a two layer hetero associative neural network.

The BAM weight matrix from the first layer to the second layer is given by

$$W = \sum_{l=1}^{L} a_l b_l^T$$

where $a_l \in \{-1,+1\}^M$ and $b_l \in \{-1,+1\}^N$ for bipolar patterns and L is the number of training patterns.

The weight matrix from the second layer to the first layer is given by

$$W^T = \sum_{l=1}^{L} b_l a_l^T$$

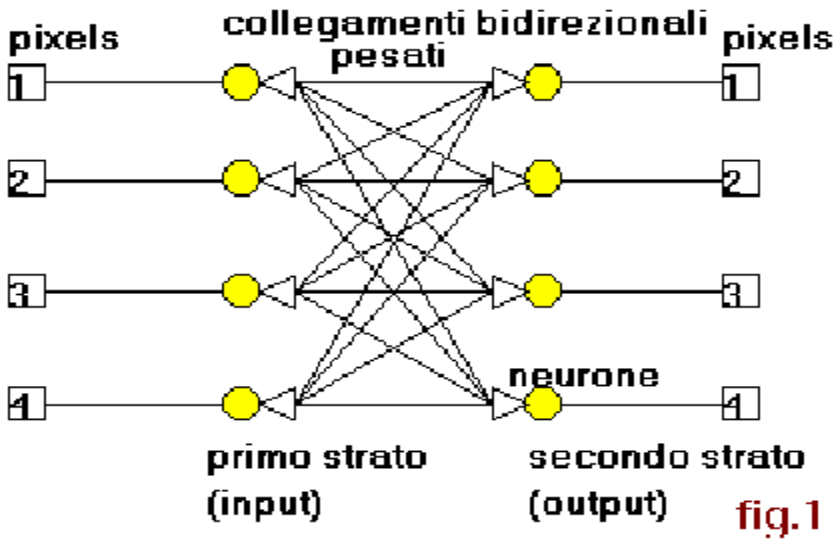


fig.1

The activation equations for the bipolar case are as follows –

$$b_j(m+1) = \begin{cases} 1 & \text{if } y_j > 0 \\ b_j(m) & \text{if } y_j = 0 \end{cases}$$

$$\begin{cases} -1 & \text{if } y_j<0 \end{cases}$$

$$\text{Where} \quad y_j = \sum_{i=1}^{M} w_{ji}a_i(m)$$

$$a_i(m+1) = \begin{cases} 1 & \text{if } x_i>0 \\ a_i(m) & \text{if } x_i=0 \\ -1 & \text{if } x_i<0 \end{cases}$$

$$\text{Where} \quad x_i = \sum_{j=1}^{N} w_{ij}b_j(m)$$

Multidirectional associative memory

The multiple association memory is also called multidirectional associative memory. Assume three layers of units denoted as A, B, C. The dimensions of the three vectors a_l, b_l, c_l are N_1 ,N_2, N_3 respectively.

The weight matrices for the pairs of layers are given by

$$W_{AB} = \sum_{l=1}^{L} a_l b_l^T \qquad W_{BC} = \sum_{l=1}^{L} b_l c_l^T \qquad W_{CA} = \sum_{l=1}^{L} c_l a_l^T$$

And

$$W_{BA} = W_{AB}^T \qquad W_{CB} = W_{BC}^T \qquad W_{AC} = W_{CA}^T$$

$$b_j(m+1) = \begin{cases} 1 & \text{if } y_j>0 \\ b_j(m) & \text{if } y_j=0 \\ -1 & \text{if } y_j<0 \end{cases}$$

For j =1, 2, …..N_2

Where

$$y_j = \sum_{i=1}^{N_1} w_{ABji}\, a_i(m) + \sum_{i=1}^{N_3} W_{CBji}\, c_i(m)$$

where w_{ABji} is the jith element of the weight matrix W_{AB} and W_{CBji} is the jith element of the weight matrix W_{CB}.

Temporal Associative Memory

The BAM can be used to store a sequence of temporal pattern vectors and recall the sequence of patterns. The basic idea is that the adjacent overlapping pattern pairs are to be stored in a BAM. Let $a_1,a_2,\ldots a_L$ be a sequence of L patterns, each with a dimensionality of M. Then $(a_1,a_2),(a_2,a_3)\ldots.(a_i,a_{i+1})\ldots(a_{L-1},a_L)$ and (a_L,a_1) form the pattern pairs to be stored in the BAM. The last pattern in the sequence is paired with the first pattern. The weight matrix in the forward direction is given by

$$W = \sum_{i=1}^{L-1} a_i a_{i+1}^T + a_L a_1^T$$

The weight matrix for the reverse direction is given by the transpose of the forward weight matrix that is W^T.