INTRODUCTION

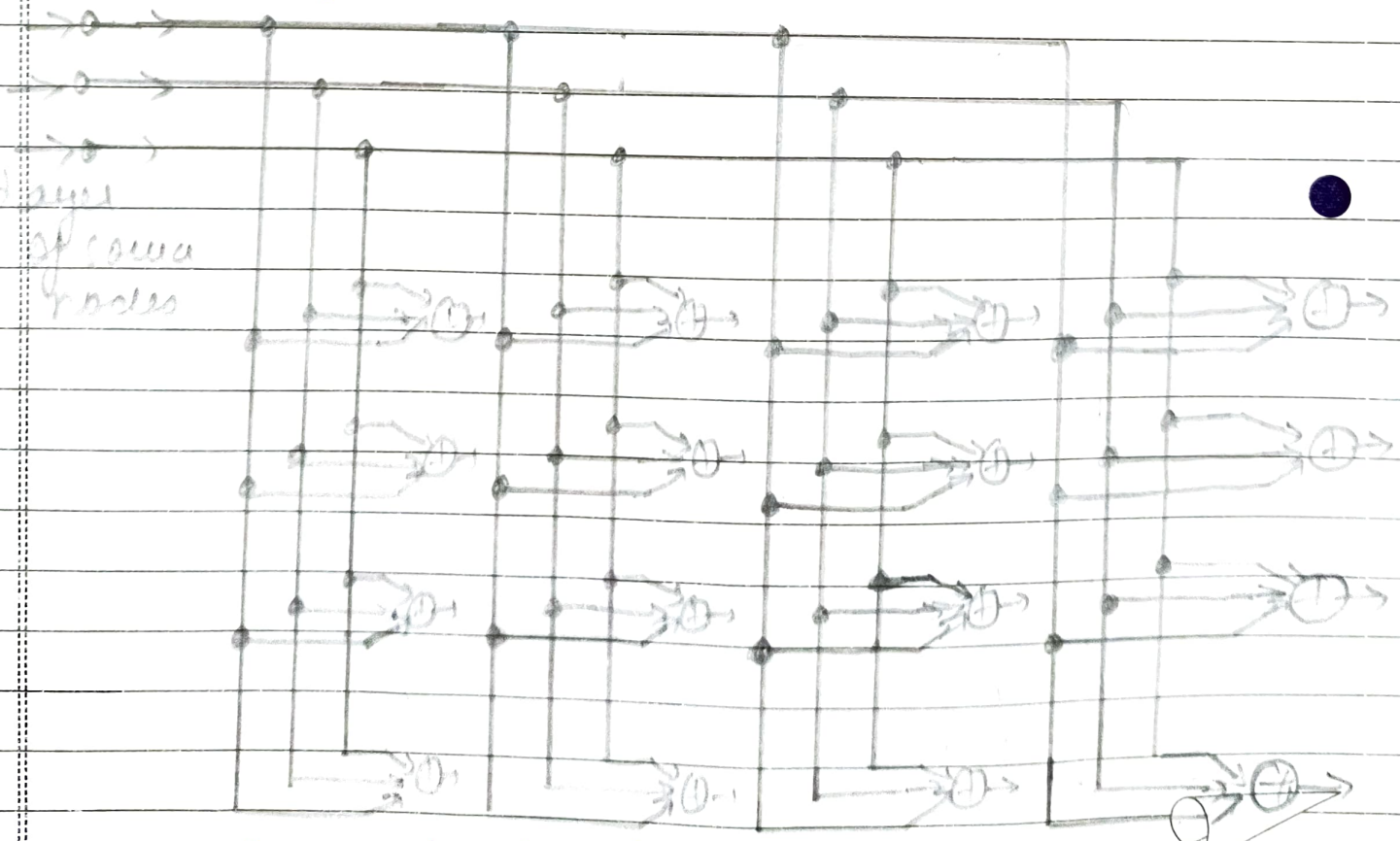
Self organising maps are based on competitive learning; the output neurons of the network compete among themselves to be activated or fired, with the result that only one output neuron, or one neuron per group, is on at any one time. An output neuron that wins the competition is called a winner-take-all neuron or simply a winning neuron.

In a ~~sff~~ self organizing maps, the neurons are placed at the nodes of a lattice that is usually one or two-dimensional. Higher dimensional maps are also possible but not as common. The neuron ~~s~~ become selectively tuned to various input patterns or class of input patterns in the course of a competitive learning process. The locations of neurons so tuned become ordered w.r.t each other in such a way that a meaningful coordinate system for different input features is created over the lattice. A self-organizing map is therefore ~~considered~~ characterized by the formation of a topographic map of the input patterns in which the spatial location of the neurons in the lattice are indicative of intrinsic statistical features contained in the input patterns, hence the name "Self organizing Maps".

As a neural model, the self organizing map provides a bridge between two levels of adaptation.

- Adaptation rules formulated at the microscopic level of a single neuron.
- Formation of experimentally better and physically accessible patterns of feature selectivity at the microscopic level of neural ~~network~~ layers.

The principal goal of the self-organizing map is to transform an incoming signal pattern of arbitrary dimension into a one- or two dimensional discrete map and to perform this transformation adaptively in a topologically ordered fashion.



2-D lattice of neurons.

Each neuron in the lattice is fully connected to all the source nodes in the input layer. This network represents a feedforward structure with a single computational layer consisting of neurons arranged in rows and columns. A one-dimensional lattice is a special case of the configuration depicted by diagram. Here computational layer consist simply of a single column or row of neurons.

COMPETITIVE LEARNING

For each input pattern, the neurons in the network compute their respective values of a discriminant function. This discriminant function provides the basis for competition among the neurons. The particular neuron with the largest value of discriminant function is declared winner of the competion.

Let m denote the dimensions of input (data) space. Let an input pattern (vector) selected at random from the input space be denoted by

$$x = [x_1, x_2, \ldots, x_m]^T$$

The synaptic weight vector of each neuron in the network has the same dimension as the input space. Let the synaptic weight vector of neuron j be denoted by

$$w_j = [w_{j1}, w_{j2}, \ldots, w_{jm}]^T, \quad j = 1, 2, \ldots, l$$

where l is the total number of neurons in the network. To find the best match of the input

vector x with the synaptic weight vectors w_j, compare the inner products $w_j^T x$ for $j = 1, 2, \ldots, l$ and select the largest. This assumes that the same threshold is applied to all the neurons; the threshold is negative of bias. Thus, by selecting the neuron with the largest inner product $w_j^T x$ we will have in effect determined the location where the topological neighborhood of excited neurons is to be centered.

The best matching criterion, based on maximizing the inner product $w_j^T x$, is mathematically equivalent to minimizing the Euclidean distance between the vectors x and w_j. If we use the index $i(x)$ to identify the neuron that best matches the input vector x, we may then determine $i(x)$ by applying the condition.

$$i(x) = \arg\min_j \|x - w_j\| \quad j = 1, 2, \ldots, l \qquad (1)$$

which sums up the essence of the competition process among the neurons. According to equation (1) $i(x)$ is the subject of attention because we want the identity of neuron i. The particular neuron i that satisfies this condition is called the best-matching or winning neuron for input vector x. Eq (1) leads to an observation-

A continuous input space of activation patterns is mapped onto a discrete output space of neurons by a process of competition among the neurons in the network.

depending on the application of interest, the response of the network could be either the index of the winning neuron or the synaptic weight vector that is closest to the input vector in a Euclidean sense.

Cooperative AND Adaptive Processes

COOPERATIVE PROCESS

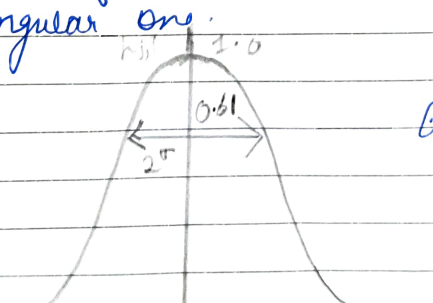
The winning neuron locates the center of a topological neighbourhood of cooperating neurons. A neuron that is firing tends to excite the neurons in its immediate neighbourhood more than those farther away from it, which is intuitively satisfying. This observation lead us to make the topological neighbourhood centered on winning neuron i decay smoothly with lateral distance. To be specific, let $h_{j,i}$ denote the topological neighbourhood centered on winning neuron i and encompassing a set of excited neurons, a typical one of which is denoted by j. Let $d_{j,i}$ denote the lateral distance between winning neuron i and excited neuron j. Then we may assume that the topological neighbourhood $h_{j,i}$ is a unimodal function of the lateral distance $d_{j,i}$ such that it satisfies two distinct requirements:

- The topological neighborhood h_{ji} is symmetric about the maximum point defined by $d_{ji} = 0$. in other words, it attains its maximum value at the winning neuron i for which the distance d_{ji} is zero.
- The amplitude of the topological neighborhood h_{ji} decreases monotonically with increasing lateral distance d_{ji} decaying to zero for $d_{ji} \to \infty$ that is a necessary condition for convergence.

A typical choice of h_{ji} that satisfies these requirements is the Gaussian function.

$$h_{ji}(x) = \exp\left(\frac{-d_{ji}^2}{2\sigma^2}\right) \quad \text{--- (1)}$$

which is translation invariant. σ is the effective width of topological neighborhood. it measures the degree to which excited neuron in the vicinity of the winning neuron participate in the learning process. In qualitative sense, the Gaussian topological neighborhood of eq (1) is more biologically appropriate than rectangular one.



Gaussian neighborhood function

For cooperation among neighboring neuron to hold, it is necessary that topological neighborhood h_{ji} be dependent on lateral distance d_{ji} b/w winning neuron i and excited neuron j in the output space rather than on some distance measure in the original input space.

This is precisely what we have in equation (1). In the case of a one-dimensional lattice d_{ji} is an integer equal to $|j-i|$. On the other hand, in the case of 2-D lattice it is defined by

$$d_{ji}^2 = \|r_j - r_i\|^2 \quad -(2)$$

where the discrete vector r_j defines the position of excited neuron j and r_i defines the discrete position of winning neuron i, both of which are measured in the discrete output space.

Another unique feature of SOM algorithm is that size of topological neighborhood shrinks with time. This requirement is satisfied by making the width σ of topological neighborhood function h_{ji} decrease with time.

$$\sigma(n) = \sigma_0 \exp\left(-\frac{n}{\tau_1}\right) \quad n = 0, 1, 2, \ldots \quad -(3)$$

σ_0 is the value of σ at initiation of SOM algorithm and τ_1 is a time constant. Correspondingly, the topological neighborhood assumes a time-varying form of its own, as shown by

$$h_{j,i(x)}(n) = \exp\left(-\frac{d_{ji}^2}{2\sigma^2(n)}\right) \quad n = 0, 1, 2, \ldots \quad -(4)$$

n = no of iteration, as n increases, $\sigma(n)$ decreases at an exponential rate & topological neighborhood shrinks in corresponding manner. $h_{j,i(x)}(n)$ is a neighborhood function.

ADAPTIVE PROCESS

The synaptic adaptive process, in the self organized formation of a feature map. For the network to be

self organizing, the synaptic weight vector w_j of neuron j in the network is required to change in relation to the input vector x. In Hebb's postulate of learning, a synaptic weight is increased with a simultaneous occurrence of presynaptic and postsynaptic activities. The use of such a rule is well suited for associative learning. For the type of unsupervised learning being considered here, however, the Hebbian hypothesis in its basic form is unsatisfactory for the following reason: changes in connectivities occur in one direction only, which finally drive all the synaptic weights into saturation. To overcome this problem we modify the Hebbian hypothesis by including a forgetting term $-g(y_j)w_j$, where w_j is the synaptic weight vector of neuron j and $g(y_j)$ is some positive scalar function of the response y_j. The only requirement imposed on the function $g(y_j)$ is that the constant term in the Taylor series expansion of $g(y_j)$ be zero, so that we may write

$$g(y_j) = 0 \quad \text{for } y_j = 0 \qquad (1)$$

The significance of this requirement will become apparent momentarily. Given such a function, we may then express the change Δw_j to the weight vector of neuron j in the network as follows:

$$\Delta w_j = \eta y_j x - g(y_j) w_j \qquad (2)$$

η is a learning-rate parameter of the algorithm. The first term on the right-hand side of eq (2) is the Hebbian term & second term is the forgetting term. To satisfy the requirement of eq (1), we choose a

linear function for $g(y_i)$ as shown by.

$$g(y_i) = \eta y_j \quad -(3)$$

we may further simplify eq (2) by using

$$y_j = h_{ji(x)} \quad -(4)$$

using eq (3) & (4) in (2) we obtain.

$$\Delta w_j = \eta h_{ji(x)} (x - w_j) \quad -(5)$$

Finally, using discrete-time formalism, given the synaptic weight vector $w_j(n)$ of neuron j at time n, the updated weight vector $w_j(n+1)$ at time $n+1$ is defined by

$$w_j(n+1) = w_j(n) + \eta(n) h_{ji(x)}(n) (x - w_j(n))$$

Vector-Quantization using SOM

Vector quantization is a technique that exploits the underlying structure of input vectors for the purpose of data compression. Specifically, an input space is divided into a number of distinct regions and for each region a reconstruction vector is defined. When the quantizer is presented a new input vector, the region in which the vector lies is first determined, and is then represented by the reproduction vector for that region. Thus, by using an encoded version of this reproduction vector for storage or transmission bandwidth can be realized at the expense of some distortion. The collection of possible reproduction vectors is called the code ~~books~~ book

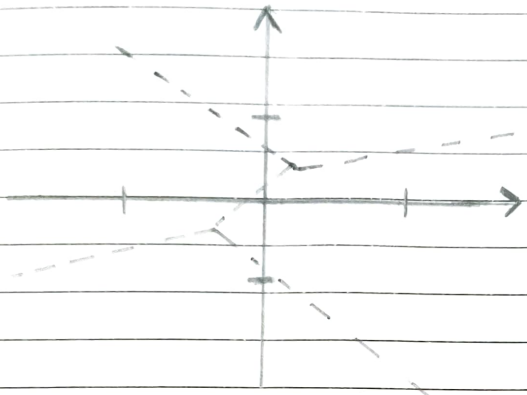
of the quantizer and its members are called code words.

A vector quantizer with minimum encoding distortion is called a Voronoi or nearest neighbour quantizer. Since the Voronoi cells about a set of points in an input space correspond to a partition of that space according to the nearest-neighbour rule based on the Euclidean metric.

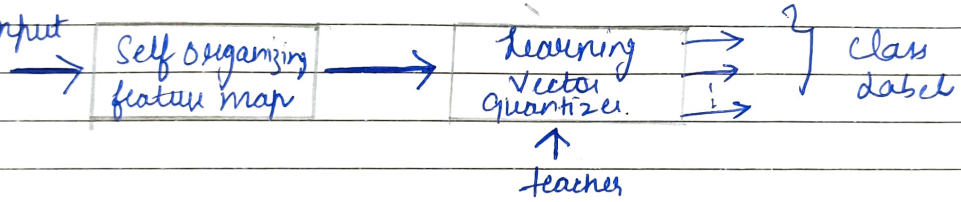
The SOM algorithm provides an approximate ~~~~ method for computing the Voronoi vectors in an unsupervised manner, with the approximation being specified by the synaptic weight vectors of the neurons in the feature map. This is merely restating property 1 of SOM algorithm. Computation of the feature map may therefore be viewed as the first of two stages for adaptively solving a pattern classification problem. The second stage is provided by learning vector quantization, which provides a mechanism for the final fine tuning of a feature map.

LVQ is a supervised learning technique that uses class information to move the Voronoi vectors slightly so as to improve the quality of classifier decision regions. An input vector x is picked at random from the input space. If the class labels of the input vector x and a Voronoi vector w agree, the Voronoi vector w is moved in the direction of input vector x. If, on the other hand, the class label of the input vector x and Voronoi vector w disagree, the Voronoi

vector w is moved away from the input vector x.



Voronoi diagram involving four cells



Let $\{w_j\}_{j=1}^{l}$ denote the set of Voronoi vectors and let $\{x_i\}_{i=1}^{N}$ denote the set of input vectors. We assume that there are many more input vectors than Voronoi vectors, which is typically the case in practice. The LVQ algorithm proceeds as follows –

(i) Suppose that Voronoi vector w_c is the closest to the input vector x_i. Let C_{w_c} denote the class associated with Voronoi w_c and C_{x_i} denote the class label of the input vector x_i. The Voronoi vector w_c is adjusted as follows:

- If $C_{w_c} = C_{x_i}$ then,
  $w_c(n+1) = w_c(n) + \alpha_n [x_i - w_c(n)]$
  where $0 < \alpha_n < 1$
- If on the other hand $C_{w_c} \neq C_{x_i}$ then
  $w_c(n+1) = w_c(n) - \alpha_n [x_i - w_c(n)]$

(iii) The other Voronoi vectors are not modified.

Maxican Hat Networks

The mexican Hat neural network is a theoretical model used in the field of computational neuro-science and neural network research. It is based on a specific type of connection pattern between neurons that resembles the shape of a Mexican hat graphically represented. This neural network model utilizes a form of lateral inhibition, where neurons both excite nearby neurons and inhibit those are further away. The term "Mexican Hat" refers to the shape of the function used to model these excitatory and inhibitory interaction, resembling a sombrero or a "hat" when visualized.

The key characteristic of the Mexican Hat Network is its spatially structured connectivity. Neurons close to each other are strongly connected and can excite each other, while those farther away have inhibitory connection, creating an activation pattern that resemble the peak and rim of a Mexican hat.

ARCHITECTURE

The neurons are attached with the excitatory positive weights to the neuron close to the reference neuron. Similarly, the neurons are also connected with

negative inhibitory weights with neuron at a farther distance. In some architecture, some neuron are further away still which are not connected in terms of weights. "All of these connections are within a particular layer of a neural net". The neural network acquires external input data apart from these connections. This data is multiplied with the corresponding weights depending on the proximity to obtain the value. ~~This will be ADD~~.

Algorithm for Mexican Hat networks –

- Let 'R_1' be the radius of reinforcement, the area closer to the reference neuron. 'R_2' be the radius of connection further away from reference neuron where $R_1 < R_2$
- w_k represents the weight of the neighbour that varies according to the proximity of the neuron. variation conditions are –

(1.) w_k is +ve if $0 < k \leq R_1$
(2.) w_k is -ve if $R_1 < k \leq R_2$
(3) w_k is zero if $k > R_2$

'x' and 'x_old' be the vector of the activations of the current and previous epoch. Finally, 't_max' be the total number of 'iterations of contrast enhancement'.

(1) Depending on the use case, initialize the values of R_1, R_2 and t_max.

now, set the initial weights of the neural network.

$w_k = c_1$ for $k = 0, 1, 2, \ldots R_1$ where $c_1 > 0$

$w_k = c_2$ for $k = R_1 + 1, R_1 + 2, \ldots R_2$ where $c_2 < 0$

(2.) For the first iteration, set the external signal vector $x = s$

Fix the activation in the x-old array for $i = 1, 2, \ldots n$ to be used in the next epoch.

(3.) Find overall input for $i = 1, 2, \ldots n$.

$(net)_i = x_i = c_1 \varepsilon (x\_old)_i + K - c_2 \varepsilon (x\_old)_i + K - c_2 \varepsilon (x\_old)_i + K.$

(4.) Apply the activation function & save current activation in x-old for using it in the next iteration.

(5.) Increase the counter variable t, as $t = t + 1$.

(6.) Test for stopping condition.

If $t < t\_max$, then continue (Go to Step 3), else stop.