MODULE - 5
SELF ORGANIZING FEATURE MAPS

Maximal Eigenvector Filtering, Extracting Principal Components, Generalized Learning Laws, Vector Quantization, Self -organizing Feature Maps, Application of SOM, Growing Neural Gas.

1. Explain the concept of dimensionality reduction using Principal Component Analysis (PCA)

1. Weight vector magnitude growing without bounds can be solved in a straightforward fashion by using PCA.
2. Eigendirections defined by the eigenvectors of the correlation matrix of input data stream provides a means of characterising the properties of a data set.
3. These are the principal component directions in the input stream that account for data's variance.
4. Given a data distribution having large deviation in x direction and very small deviation in y direction, x direction is the dominant direction and no impact happens if we were to retain only x components and sacrifice y components. X components would be able to reconstruct a data set within a certain maximum error.
5. In higher dimensional applications, we can discard less important directions and keep only important ones.
6. High dimension data can be effectively reduced onto lower dimension while retaining most of the essential intrinsic information required for reconstruction of data points.
7. But this leads to large economic costs and losses.
8. So it's important to be able to identify principle characterising components for sufficiently accurate reconstruction of data.
9. PCA essentially involves analysis of data to find directions along which variations are extremal.
10. The theory attempts to search for an effective mapping of data points from original n dimensional space onto lower m dimensional space.
11. This involves finding m dominant directions in input space (m<n) with the aim of prospecting n data to m subspace spanned by these m principal components, without sacrificing the information content of the data. This is called dimensionality reduction.
12. For 0 mean data, first m principle components are those eigenvectors of matrix R, that correspond to m largest eigenvalues of R.
    Given eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_m$ and directions γ_i

    $X = \sum_{i=1}^{n} \gamma_i \eta_i$ m<n
13. This means neglect coefficients with small eigenvalues and retain coefficients with large eigenvalues.
14. Therefore to reduce dimension, variance of error σ_e^2

$$\sigma_e^2 = \sum_{i=m+1}^{n} \lambda_i$$

i) Analyze R (correlation matrix) of data steam and find its eigenvalues and eigenvectors.
ii) Project data onto eigendirections.
iii) Discard n-m components corresponding to n-m smallest eigenvalues.

2. What is Vector Quantization and describe the architecture of supervised and unsupervised vector quantisation.

Vector quantization is a technique that exploits the underlying structure of input vectors for the purpose of data compression. It works by dividing a large set of points (vectors) into groups having approximately the same number of points closest to them. Each group is represented by its centroid point.

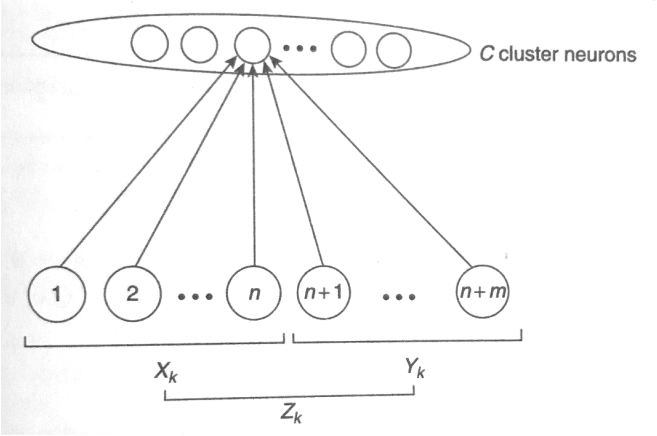
1. An input space is divided into a number of distinct regions and for each region, a reconstruction vector is defined. These regions are called Voronoi regions which are later identified as cluster classes or decision regions.
2. Each region is represented by a single vector called codebook vector.
3. When the quantiser is presented with a new input vector, the region on which the vector lies is determined and represented by that codebook vector.
4. Meaning every point in input space belongs to one of these regions and is mapped to corresponding or nearest codebook vectors.
5. Compression is achieved by identifying a signal vector in a region of input space by a codebook vector. Then all vectors falling to that region are represented by that codebook vector.
6. In other words, a set of codebook vectors is a compressed form of a set of codebook vectors is a compressed form of a set of input data vectors as many input vectors can be mapped to the same codebook vector. Eg: Voronoi tesellations that depict classification regions formed by 1 NN using minimum encoding.
7. The aim in vector quantisation is that codebook vectors w_i are required to be placed in signal space in a way that minimizes average expected quantisation error.

   $E = \int ||X-W_j||^2 \, p(X) \, dX$

   Where W_j = winning codebook vector.

Unsupervised VQ :

1. If there are Q input output pairs $\{ X_k , Y_k \}$ of data that need to be clustered, they are concatenated to form vector $Z_k = (X_k, Y_k)$ then employed in an autoassociative VQ learning process.



2. So these cluster neurons compete for activation from Z_k which is now the concatenated input pattern/vector.
3. The unsupervised VQ system compare current sample z_k with cluster C weight vector w_j and I is the winner based on minimum Euclidean distance as :
   $||W_{j(k)} - Z_k|| = \min ||W_{j(k)} - Z_{(k)}||$
4. Winner J is based on competitive learning where each neuron competes to be the one active neuron that undergoes weight updation.
5. Distance from each z_k to $w_{j(k)}$ is taken and minimum is the winner.
6. Winning neuron J learns the input pattern by competitive learning. In vector form, given weight update equation
   $N_{j(k+1)} = W_{j(k)} + \eta_k (Z_{(k)} - W_{j(k)})$
7. We scale the components of data samples z_k so all features have equal weight during distance measure.
8. $$(W_{J(k)} - Z_k)^T \Omega (W_{J(k)} - Z_k) = \min_j \left\{ (W_{j(k)} - Z_k)^T \Omega (W_{j(k)} - Z_k) \right\}$$
   where
   $$\Omega = \begin{bmatrix} \omega_1^2 & 0 & \dots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \omega_{n+m}^2 \end{bmatrix}$$
   This is the data dependent scaling matrix.
   (scaling ensures no variable dominates the winner)

Supervised VQ :

1. This is also called learning vector quantisation.
2. Here data classes are defined in advance such that each data sample is labelled with its class.
3. It applies to the winner take-all learning.
4. It illustrates how unsupervised learning can be adapted to solve supervised learning tasks where classes are known.
5. Each node is associated with arbitrarily chosen class label
6. We keep no. of nodes for each class same as no. of training pattern for class. So that each cluster has the same no. of patterns.
7. For a neuron assumed to be labelled as class C_j the weight update procedure for winning neuron J and other j not equal to J is given as :

For j=J

$W_{iJ}^{k+1} = W_{iJ}^{k} + \eta_k(X_i^k - W_{iJ}^{k})$ X_k belongs C_j

$W_{iJ}^{k} - \eta_k(X_i^k - W_{iJ}^{k})$ X_k doesn't belongs C_j

For $j \neq J$

$W_{iJ}^{k+1} = W_{iJ}^{k}$ $j \neq J$

8. We aim to cyclically apply this till η_k kept small.
9. Difference between supervised and unsupervised VQ is that if vectors are misclassified, those weight vectors are moved away from present input to reduce misclassification rather than keep them the same for the next iteration.

3. What is Self - Organising Featuer Maps (SOFM) and what are its operational details.

1. Self Organisation Feature Maps showcases dimensionality reduction with the preservation of topological information.
2. We compress information by extracting relevant facts and develop reduced representation of implying information while retaining essential knowledge .
3. It produces simplified internal representation of the external world with different levels of abstraction.
4. The aim is to identify or extract a set of features (invariate) that concentrate essential information of the input pattern set.
5. Kohenens derived this approach where important topological information can be obtained through supervised learning .
6. These feature maps are designed for visualisation, compression and clustering.
7. Being a competitive learning VQ approach, patterns are presented sequentially to linear/ planar arrays of neurons that interact like Mexican hats.
8. These interactions allow clusters of neurons to win the competition rather than just one neuron.
9. Then weights of these winning clusters are adjusted to bring better response to current input.
10. This competition- adaptation process when done iteratively will result in weights showing clusters that are topologically close (not physically close in input space) .
11. There is a correspondence between signal features and response locations on a map.
12. SOFM preserves this topology of input,
13. So it transforms incoming signal patterns of arbitrary dimension to one or two dimensional discrete maps.

<u>Operational details of SOFM :</u>

For topological preservation of information, distance relation between high dimensional space should be approximated by network as distance in 2D NN. For this mapping,

1. Input neurons should be exposed to a sufficient number of different inputs.
2. For a given input, only the winning neuron and neighbour adapts weights.

3. Do this to all adjacent neurons that are topologically related subsets.
4. Resulting adjustment should enhance the responses to the same or similar input that occurs subsequently.

Assume the input vector $X_k = (x_1^k, \ldots x_n^k)^T$ belongs to R^n is presented to the mxm neuron field. Each neuron is identified by row - column i,j.. I,j = 1…m . The ij th neuron has incoming vector $W_{ij(k)} = (w_{1j} \ldots w_n^k{}_j)$

1. First find the best matching weight vector $W_{1j(k)}$ for the present input and identify the neighbourhood around N_{ij} the winning neuron.
2. Best matching neuron is found by composing the inner product $X_k^T W_{ij}$ with X_k and each W_{ij} .
3. The largest inner product is the winning neuron because larger inner product means least Euclidean distance.
   $||X_k - W_{ij(k)}|| = \min ||X_k - W_{ij(k)}||$
4. The neighbourhood of W_{ij} is identified by radius measured discreetly by each Mexican Hats interaction.
5. As more epochs elapse, the neighbourhood shrinks.
6. Once winning cluster is identified, the weights within the clusters are updated :

$$W_{ij} = \begin{cases} \eta\ (X - W_{ij}) & i,j \text{ belongs } N_{ij} \\ 0 & i,j \text{ doesn't belongs } N_{ij} \end{cases}$$

$$W_{ij(k+1)} = \begin{cases} W_{ij(k)} + \eta_k (X_k - W_{ij(k)}) & i,j \text{ belongs } N_{ij} \\ W_{ij(k)} & i,j \text{ doesn't belongs } N_{ij} \end{cases}$$

4. Discuss two applications of SOFM

PATTERN CLASSIFICATION

1. Feature maps have each neuron labellings with a test pattern that has maximally excited the neuron.
2. This produces a well ordered partition of map such that clusters will respond to the same class of pattern. This creates similarity relationships used for pattern classification.
3. In the iris dataset, components are normalised so that all have unity variance (no dominance over the other).
4. Assume an 8 x 8 grid labelled with classes.
5. Since SOFM uses Euclidean distances, the scale of individual variables plays an important role in how the final map will be.
6. We normalise the components of data so that there is no dominance over the other.
7. The best matching neuron is the one with minimum distance between weight vector and pattern under consideration.
8. The species label corresponding to the pattern is assigned to the neuron.
9. All neurons will then have a label assigned that corresponds to species whose patterns elicited a maximal response with highest frequency.

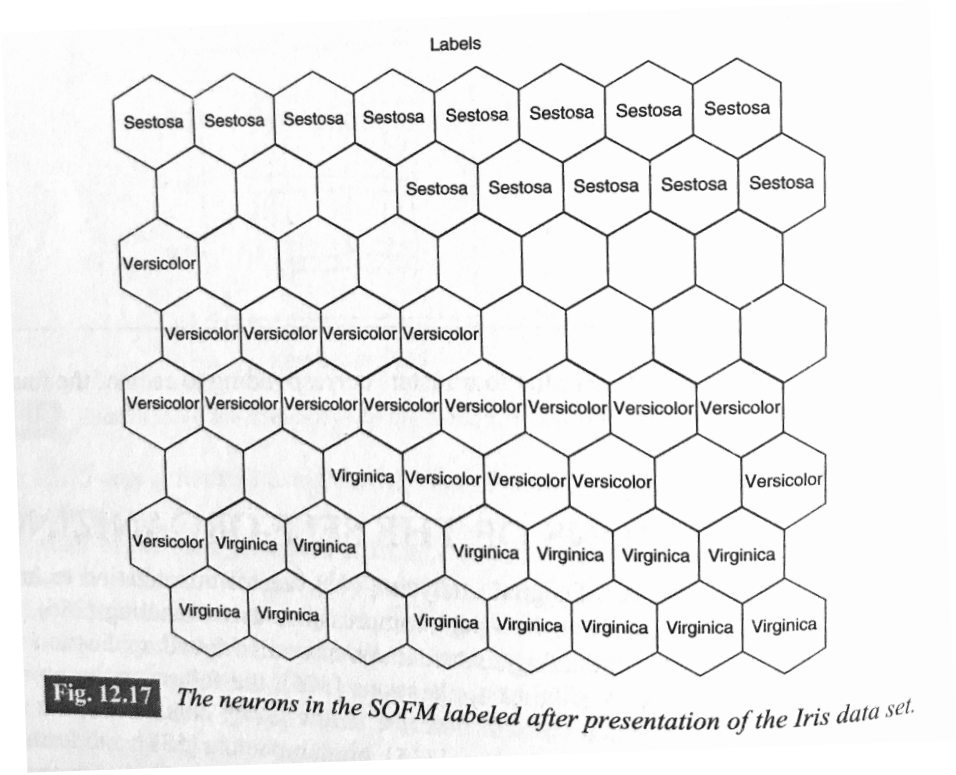
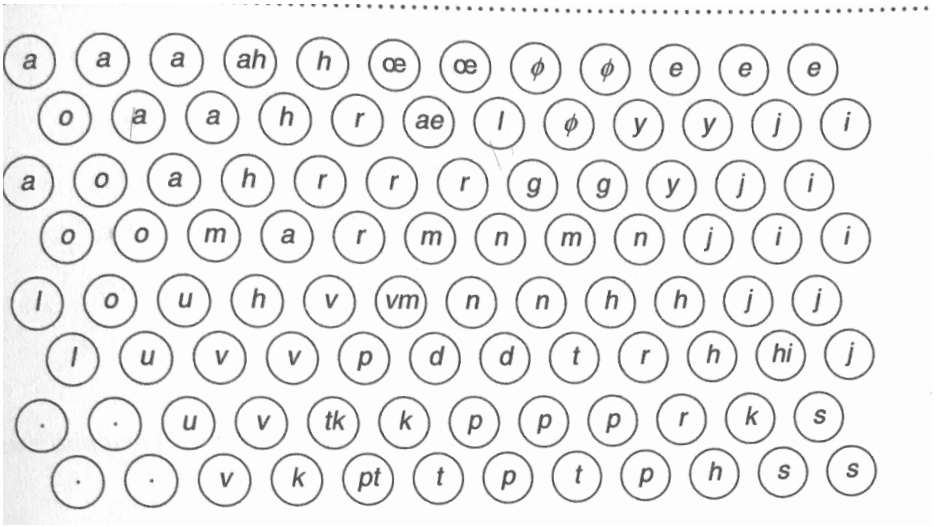


Fig. 12.17 *The neurons in the SOFM labeled after presentation of the Iris data set.*

NEURONAL PHONETIC TYPEWRITER

1. Kohenen's neural phonetic typewriter represents a classic example of an automated speech recognition that uses one of two neural network modules - SOFM and LVQ.
2. In speech recognition, there are phonal sounds broken down to simple phonemes.
3. These phonemes correspond to different linguistic units.
4. The problem is that phonemes have varying lengths, overlap partly and vary speaker by speaker.
5. The objective is to transcribe spoken utterances into phoneme sequences.
6. This typewriter works with Finnish language having 21 phonemes which corresponds to unique graphemes (text symbols)
7. The equipment was a coprocessor board that has the capacity to perform the classification of short time speech spectra by SOFM and apply symbolic postprocessing to correct for coarticulation errors. It did this with 92% accuracy.
8. The input vector uses 15 spectrum vector Xs after processing. Each input vector is tagged to a class using output Xk which is a binary vector 0 or 1.
9. Since spectra of different classes overlap, we concantenate Xk vector $S = [\ Xs^T\ Xu^T]$. This X is used as input.
10. The weight vectors are Wij approximates density of X.
11. Since Xu is now the same for vectors of the same class , but different for different classes, the clustering of X along classes is enchanted and so is the class separation.
12. Since phonemes are influenced by neighbouring phonemes, SOM are used for clustering.



5. Write a short note on Growing Neural Gas (GNG)

1. For SOFM or VQ, since input size is not always known, it is different to determine apriori the no. of neurons to use.
2. GNG makes no assumptions on no. of neurons to be used by the network.
3. It is an unsupervised incremental clustering algorithm that can even perform both dimensionality reduction while discovering a topological map of input data distribution.
4. It incrementally creates a network of nodes that quantize an input space described by distribution of points R^n drawn from unknown probability density function P(X).
5. The position vectors in GNG act as codebook vectors for all data centers and Neural Gas (NG) is used as a VQ quantiser with principle for each X_k belonging to R^n , adapt position vectors of m closest nodes in a graph with m decreasing from large initial to small final value.
6. Nodes are added incrementally during execution and finally all nodes move towards areas in input space where $p(x) > 0$.

<u>Steps :</u>

1. Scatter two nodes a and b at random positions X_a and X_b in R^n .
2. Generate an input X_k iteration k in accordance with P(X).
3. Find the nearest node n1 and second nearest node n2 using Euclidean distance.
4. Add the square distance between input $X_k$and nearest node n1 in input space to a node local error variable.

   $E_{n1\,(k+1)} = E_{n1(k)} + ||X_{n1(k)} - X_k||^2$
5. Move n1 and its direct topological neighbours N_{n1} towards X_k with leaving rate η_w and η_N of distance from each to X_k .

   $A_{n1(k+1)} = X_{n1(k)} + \eta_w(X_k - X_{n1(k)})$

   $A_{f(k+1)} = X_{j(k)} + \eta_N(X_k - X_{j(k)})$
6. Increment the age of all edges connected to n1.
7. Connect n1 and n2 by an edge : if an edge exists , set the age of this edge to zero. If such an edge does not exist, then create it and set it to zero.

8. Remove edges with age larger than A_{max}. If nodes have no edges connecting them, then remove nodes.
9. If the no. of input vectors generated so far is an integer multiple of a parameter λ , insert a new node into the network as follows :
    a) Identify node p with max accumulated error.
    b) Identify node q which is neighbour p with largest error.
    c) Insert new node r halfway between p and q :
    $X_r = 0.5 (X_p + X_q)$
    d) Insert edges connecting the new node r with nodes p and q and remove the original edge between p and q.
    e) Decrease the error of p and q by a factor of A. Initialise the local error variable of r to the new value of error variable of p.
10. Decrease all node error variables by a factor B.
11. If a stopping criterion is not reached , return to step 1.