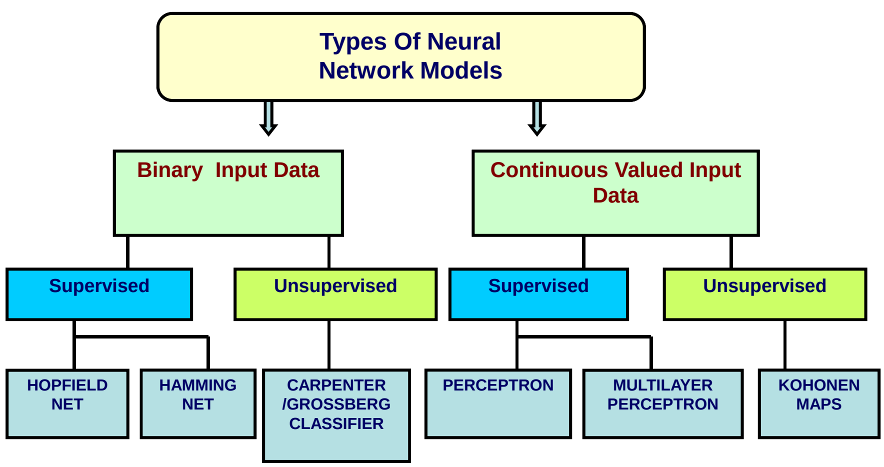
# Self Organizing Maps

Lecture 12

1

## Neural Models




2

## *An Example of Competitive Learning*

- **Goal** : To construct a neural network where each of the output neurons fires under a class of input patters

3

## *Structure*

- Output (post-synaptic) neurons are organized in a spatial pattern (typically 1-D, 2-D or 3-D) corresponding to spatial characteristics of the problem domain

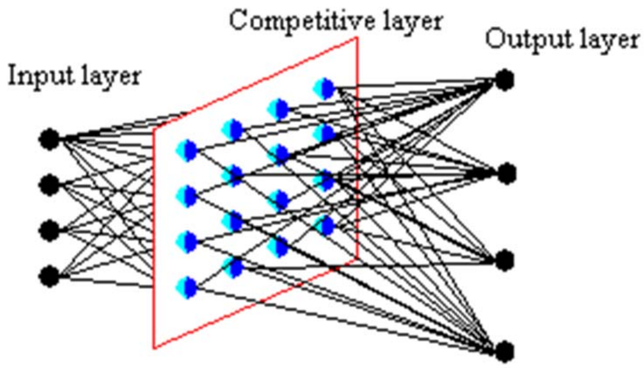
- The learning process has **TWO** steps

✓ A specialization step where each neuron is trained to a specific class of inputs

✓ A re-organization step where the neurons of the output layer are "placed" so they correspond spatially to the problem domain

4

## Network Architecture






## Input Layer

- accepts multidimensional input pattern from the environment
- an input pattern is represented by a vector.
- e.g. a sound may consist of pitch, background noise, intensity, etc.
- each neuron in the input layer represents one dimension of the input pattern
- an input neuron distributes its assigned element of the input vector to the competitive layer

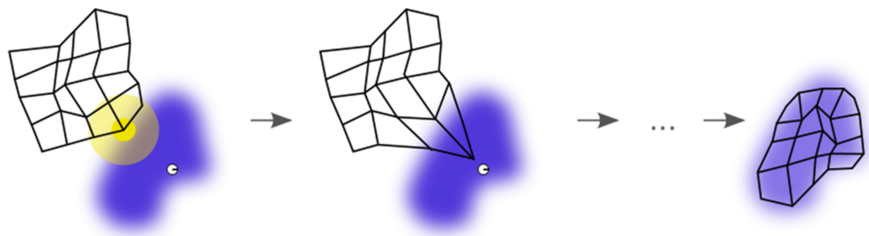
## Competitive Layer

- each neuron  in the competitive layer receives a sum of weighted inputs from the input layer
- every neuron in the competitive layer is associated with a collection of other neuron which make up its 'neighbourhood'
- competitive Layer can be organized in 1 dimension, 2 dimensions, or ... n dimensions
- typical implementations are 1 or 2 dimensions.
- upon receipt of a given input, some of the neuron will be sufficiently excited to fire. this event can have either an inhibitory, or an excitatory effect on its neighbourhood



## Output Layer

- organization of the output layer is application-dependent



9

## *Learning Process*

- Competitive Step
- Cooperation Step
- Synaptic modification step
- Convergence Step

10

## Competitive Step

- For each of the output neurons, compute the output
- Declare the neuron with the maximum output to be the winner



## Cooperative Step

- Define a topological neighborhood around the winning neuron
- The topological neighborhood can be defined using any reasonable ‘closeness’ measure
- A Gaussian distribution is typically used for the proximity measure
- The neighborhood reach should be diminished through the iterations of the learning process

## Synaptic Modification Step

- For all neurons in the neighborhood of the winning neuron, modify the synaptic weights according to the rule

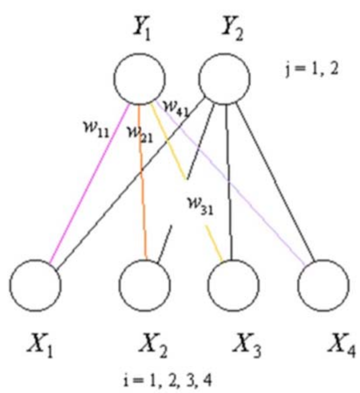
  - ***For Learning Rule details refer – Haykin's Book***



## Self Organizing Map - Example

| Step | Action |
|---|---|
| 0 | Initialize weights. Set max value for $R$, set learning rate $\alpha$. |
| 1 | While stopping condition false do steps 2 to 8 |
| 2 | For each input vector $\underline{\mathbf{x}}$ do steps 3 to 5 |
| 3 | For each j neuron, compute the **Euclidean** distance $D(j) = \sqrt{\sum_{i=1}^{n}(x_i - w_{ij})^2}$ |
| 4 | Find the index $J$ such that $D(J)$ is a minimum |
| 5 | For all neurons $j$ within a specified neighbourhood of J and for all $i$ $w_{ij}(new) = w_{ij}(old) + \alpha(x_i - w_{ij}(old))$ |
| 6 | Update learning rate $\alpha$. It is a decreasing function of the number of epochs. |
| 7 | Reduce radius of topological neighbourhood at specified times |

To make the problem very simple, suppose that there are only two neurons in the output layer as shown below:




Let the initial weight matrix be

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.5 & 0.7 \\ 0.9 & 0.3 \end{bmatrix}$$

| $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |

Consider a simple example in which there are only **4 input training patterns**.

Let the learning rate at time $t+1$ be given by $\alpha(t+1) = \dfrac{\alpha(t)}{2}$,

and suppose $\alpha(t=0) = 0.6$

Let topological radius $R = 0$.

Following the algorithm presented in the previous algorithm:

For vector 1100 *(We are using the Euclidean distance squared for convenience)*

$D(1) = (1-0.2)^2+(1-0.6)^2+(0-0.5)^2+(0-0.9)^2 = 1.86$

$D(1) = 1.86$, $D(2) = 0.98$

Hence J = 2. Note that $R = 0$, so we need not update the weights of any neighboring neurons.

Using $w_{ij}(new) = w_{ij}(old) + \alpha(x_i - w_{ij}(old))$, the new weight matrix is

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.92 \\ 0.6 & 0.76 \\ 0.5 & 0.28 \\ 0.9 & 0.12 \end{bmatrix}$$

$w_{12}(new) = w_{12}(old) + \alpha(x_1 + w_{12}(old))$
$= 0.8 + 0.6(1-0.8) = 0.92$

*Likewise for remains training patterns*

For vector 0001

D(1) = 0.66, D(2) = 2.2768

Hence J = 1.

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.92 \\ 0.24 & 0.76 \\ 0.20 & 0.28 \\ 0.96 & 0.12 \end{bmatrix}$$

For vector 0011

D(1) = 0.7056, D(2) = 2.724

Hence J = 1

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.032 & 0.968 \\ 0.096 & 0.304 \\ 0.680 & 0.112 \\ 0.984 & 0.048 \end{bmatrix}$$

For vector 1000

D(1) = 1.8656, D(2) = 0.6768

Hence J = 2

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.968 \\ 0.24 & 0.304 \\ 0.20 & 0.112 \\ 0.96 & 0.048 \end{bmatrix}$$



Now reduce learning rate (step 6): $\alpha(1) = \frac{\alpha(0)}{2} = \frac{0.6}{2} = 0.3$

It can be shown that after **100 presentations** of all the input vector, the final weight matrix is

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 6.7x10^{-17} & 1 \\ 2x10^{-16} & 0.49 \\ 0.51 & 2.3x10^{-16} \\ 1 & 1x10^{-16} \end{bmatrix}$$

This matrix seems to converge to

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0.5 \\ 0.5 & 0 \\ 1 & 0 \end{bmatrix}$$

Cluster 1 Cluster 2

**TEST NETWORK**

Suppose the input pattern is 1100. Then

$$D(j) = (w_{1j} - x_1)^2 + (w_{2j} - x_2)^2 + (w_{3j} - x_3)^2 + (w_{4j} - x_4)^2$$

$$D(1) = (0-1)^2 + (0-1)^2 + (0.5-0)^2 + (1-0)^2 = 3.25$$

$$D(2) = (1-1)^2 + (0.5-1)^2 + (0-0)^2 + (0-0)^2 = 0.25$$

This matrix seems to converge to

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0.5 \\ 0.5 & 0 \\ 1 & 0 \end{bmatrix}$$

Cluster 1 Cluster 2

Thus neuron 2 is the "**winner**", and is the localized active region of the SOM. Notice that we may label this input pattern to belong to cluster 2.

For all the other patterns, we find the clusters are as listed below.

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | Cluster |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 2 |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 2 |
| 0 | 0 | 1 | 1 | 1 |

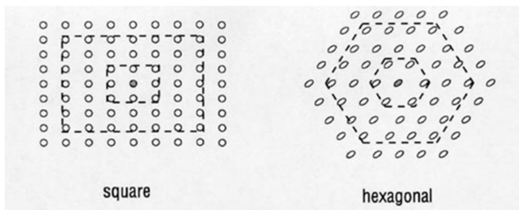
7 / 1:45 Scroll for details

## *Convergence Step*

- Fix the learning rate and the variance at the final values obtained from the adaptation step
- Continue the training process until reasonable convergence is achieved.

# Convergence

- Decreasing the neighbor ensures progressively finer features are encoded
- gradual lowering of the learn rate ensures stability

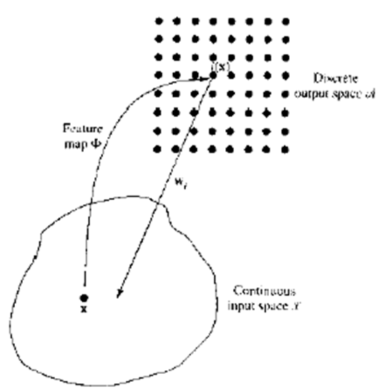


Kohonen neighbourhoods

23

# Feature Mapping

- A Mapping from continuous input space to a discrete output space.




24

## Feature Map Properties

- Approximation of input space
- Topological ordering
- Density Matching
- Feature Selection

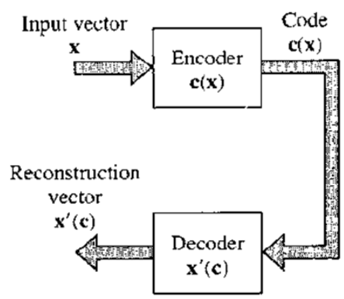
25

## Approximation of Input Space

- The SOM mapping is an approximation of the input space
- Desirable property: Dimensionality reduction
- If a perfect decoder of the encoded signal is to be designed, reconstructed vector will involve some error

26

## Approximate of Input Space




$$Error = \| x - x' \|$$



## Topological Ordering

- The spatial distribution of the output neurons topologically corresponds to the different features of the input space
- Logically, this is equivalent to physically moving the neurons to correspond to the input space.

## *Density Matching*

- The density of the output neurons matches the probability distribution of the input space, i.e. there are more output neurons representing regions where there are more training vectors resulting in better accuracy for these regions
- If there are huge variations of input density, very low density areas will be over-represented, and very high density areas will be under-represented.



## *Feature Selection*

- SOM networks extract the principal features of the input space
- SOM networks can be viewed as a generalization of Principal Components Analysis to include non-linear principal features.

# Example 1: No Dimensionality Reduction

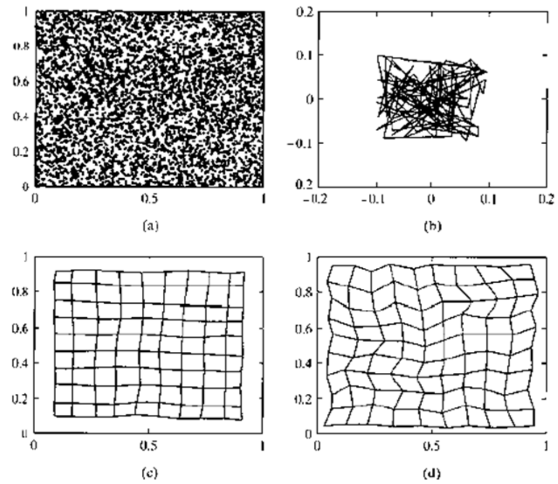


FIGURE 9.8 (a) Input data distribution. (b) Initial condition of the two-dimensional lattice. (c) Condition of the lattice at the end of the ordering phase. (d) Condition of the lattice at the end of the convergence phase.

31

# Example 2: Dimensionality Reduction

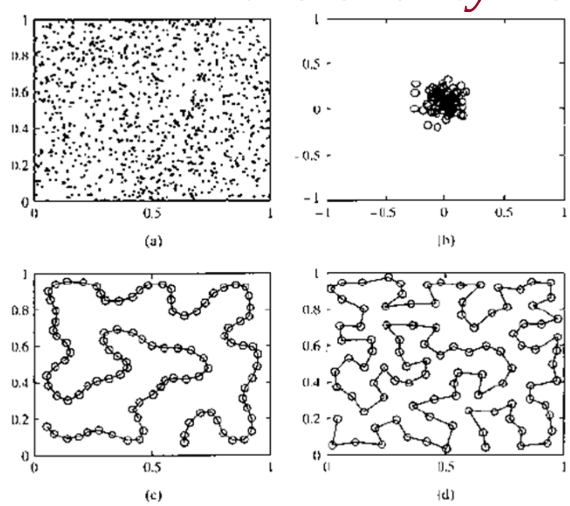


FIGURE 9.9 (a) Two-dimensional input data distribution. (b) Initial condition of the one-dimensional lattice. (c) Condition of the lattice at the end of the ordering phase. (d) Condition of the lattice at the end of the convergence phase.

32

## *Example 3: Classification*

- Starting with a set of Animals and Characteristics, train a network to extract the relevance "features"

33

## *Example 3: Inputs*

**Animal Names and Attributes**

| Animal | | Dove | Hen | Duck | Goose | Owl | Hawk | Eagle | Fox | Dog | Wolf | Cat | Tiger | Lion | Horse | Zebra | Cow |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| is | small | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | medium | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | big | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| has | 2 legs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 legs | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | hair | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | hooves | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | mane | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| | feathers | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| likes to | hunt | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| | *fly* | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | swim | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

34

## Example 3:
## Trained Network

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| dog | dog | fox | fox | fox | cat | cat | cat | eagle | eagle |
| dog | dog | fox | fox | fox | cat | cat | cat | eagle | eagle |
| wolf | wolf | wolf | fox | cat | tiger | tiger | tiger | owl | owl |
| wolf | wolf | lion | lion | lion | tiger | tiger | tiger | hawk | hawk |
| wolf | wolf | lion | lion | lion | tiger | tiger | tiger | hawk | hawk |
| wolf | wolf | lion | lion | lion | owl | dove | hawk | dove | dove |
| horse | horse | lion | lion | lion | dove | hen | hen | dove | dove |
| horse | horse | zebra | cow | cow | cow | hen | hen | dove | dove |
| zebra | zebra | zebra | cow | cow | cow | hen | hen | duck | goose |
| zebra | zebra | zebra | cow | cow | cow | duck | duck | duck | goose |

**Three Clusters - Birds, Peaceful Species, and hunters**

35

## *Some More Examples*

36

## *Letter and Word recognition*

*Rumelhart & Zipser (1986)*

- Training set {AA, AB, BA, BB}
  - 2 units learn to detect either A or B in an particular serial position
  - 4 units learn the pairs : word detector
- Training set {AA, AB, AC, AD, BA, BB, BC, BD}
  - 2 units learn to respond pairs start with A or B
  - 4 units learn to recognize pairs end with A, B, C, or D
  - 8 units learn to learn the pairs

37

## *2D Map of 3D Region*

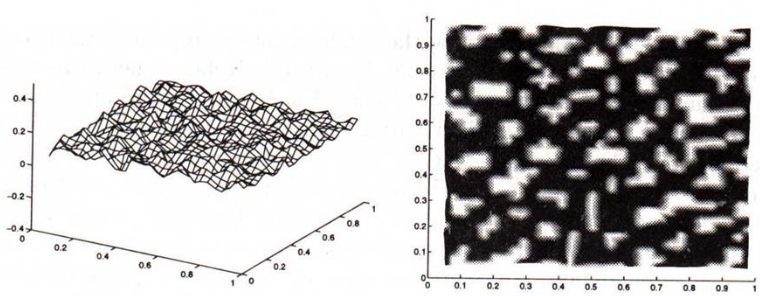
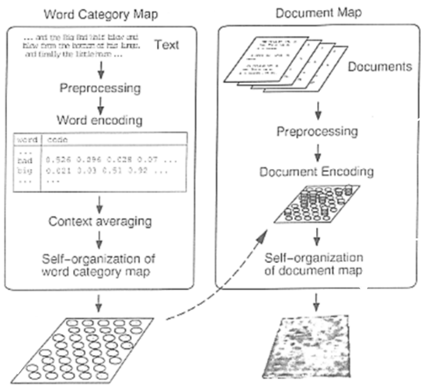


**Fig. 15.9.** Two-dimensional map of a three-dimensional region

38

# *WEBSOM*




All words of document are mapped into the word category map

Histogram of "hits" on it is formed

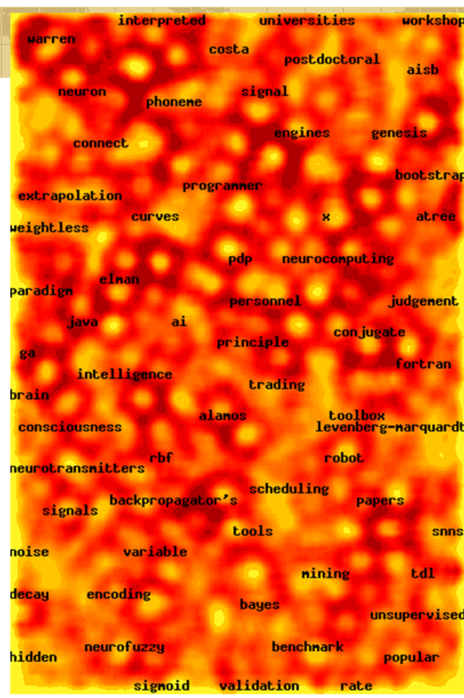
**Self-organizing semantic map.**
**15x21 neurons**
**Interrelated words that have similar contexts appear close to each other on the map**

**Self-organizing map.**
Largest experiments have used:

- word-category map
  315 neurons with 270 inputs each
- Document-map
  104040 neurons with 315 inputs each

- Training done with 1124134 documents

39




40

41