Introduction

The main property of a neural network is an ability to learn from its environment, and to improve its performance through learning. So far we have considered **supervised** or **active learning** – learning with an external "teacher" or a supervisor who presents a training set to the network. But another type of learning also exists: **unsupervised learning**.

Introduction

- Unsupervised learning
  - Training samples contain only input patterns
    - » No desired output is given (teacher-less)
  - Learn to form classes/clusters of sample patterns according to similarities among them
    - » Patterns in a cluster would have similar features
    - » No prior knowledge as what features are important for classification, and how many classes are there.

Introduction

- NN models to be covered
  - **Hebbian learning**
    - » **Generalised Hebbian learning algorithm**
  - Competitive networks
    - » Self-organizing map (SOM)
- Applications
  - Clustering
  - segmentation
  - Feature extraction
  - optimization

- In contrast to supervised learning, unsupervised or **self-organised learning** does not require an external teacher. During the training session, the neural network receives a number of different input patterns, discovers significant features in these patterns and learns how to classify input data into appropriate categories. Unsupervised learning tends to follow the neuro-biological organisation of the brain.
- Unsupervised learning algorithms aim to learn rapidly and can be used in real-time.

Competitive learning

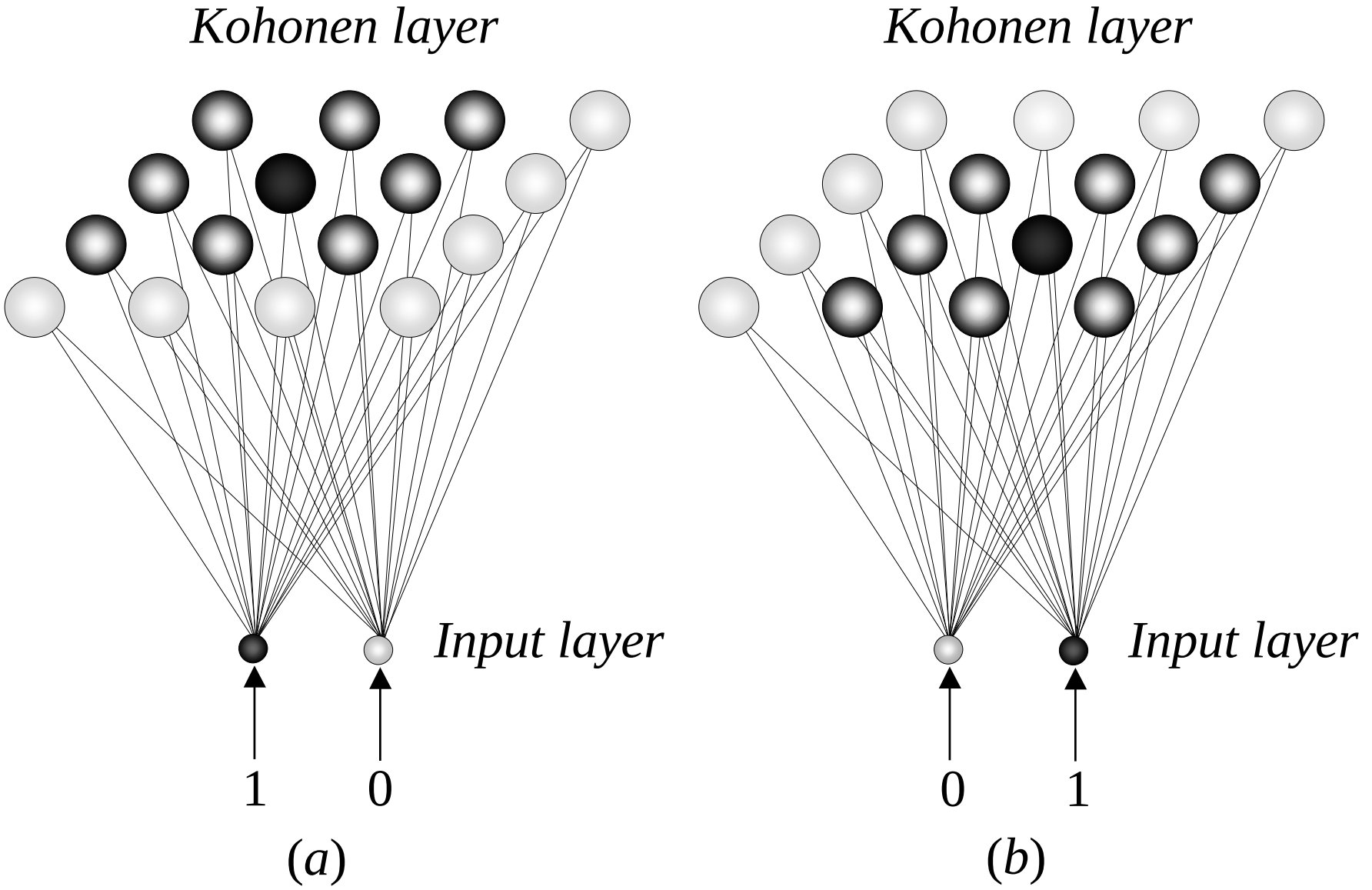
- In competitive learning, neurons compete among themselves to be activated.
- While in Hebbian learning, several output neurons can be activated simultaneously, in competitive learning, only a single output neuron is active at any time.
- The output neuron that wins the "competition" is called the ***winner-takes-all*** neuron.

- The basic idea of competitive learning was introduced in the early 1970s.
- In the late 1980s, Teuvo Kohonen introduced a special class of artificial neural networks called **self-organising feature maps**. These maps are based on competitive learning.

What is a self-organising feature map?

Our brain is dominated by the cerebral cortex, a very complex structure of billions of neurons and hundreds of billions of synapses. The cortex includes areas that are responsible for different human activities (motor, visual, auditory, etc.), and associated with different sensory inputs. We can say that each sensory input is mapped into a corresponding area of the cerebral cortex. **The cortex is a self-organising computational map in the human brain.**

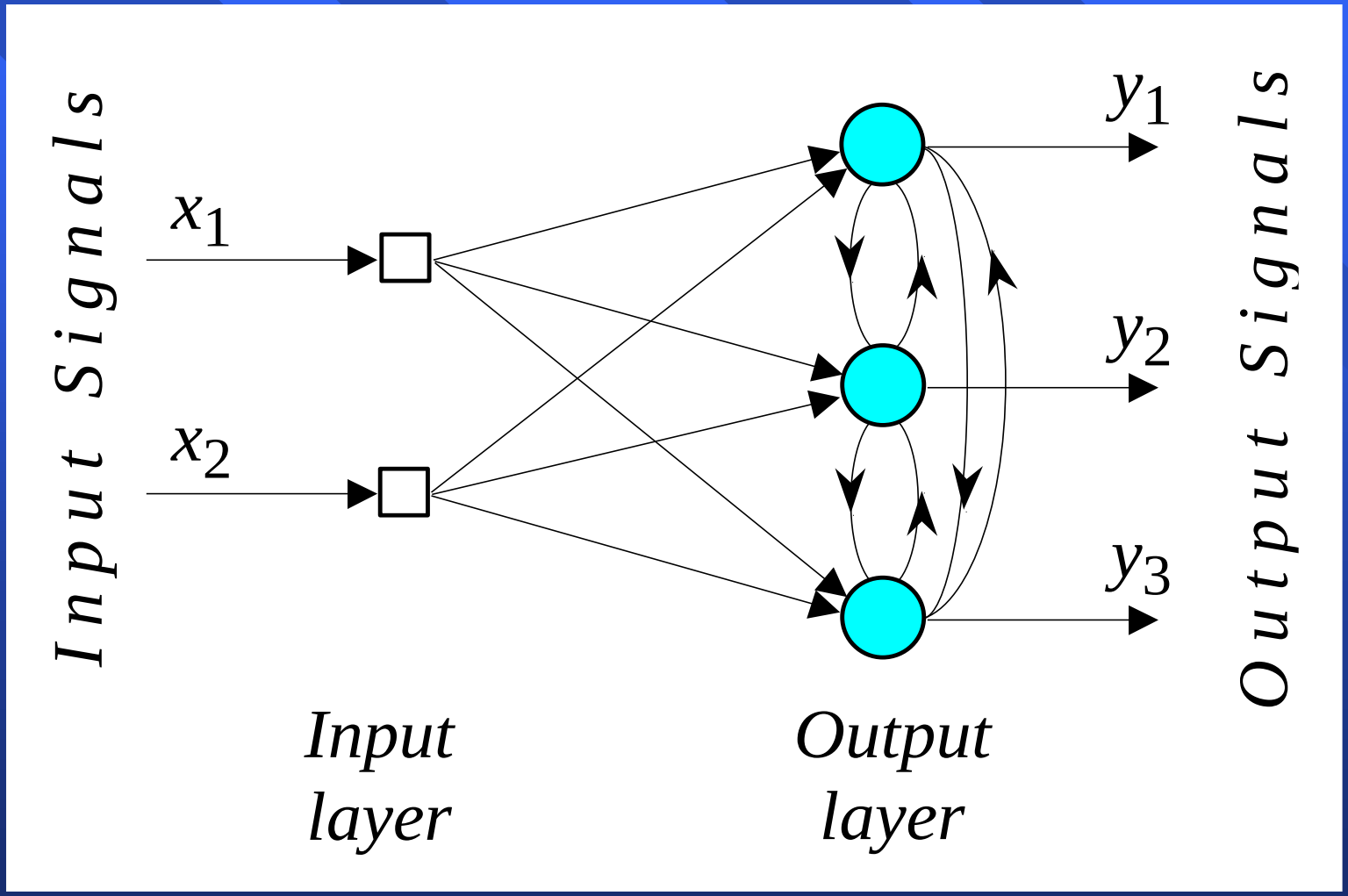
Feature-mapping Kohonen model



The Kohonen network

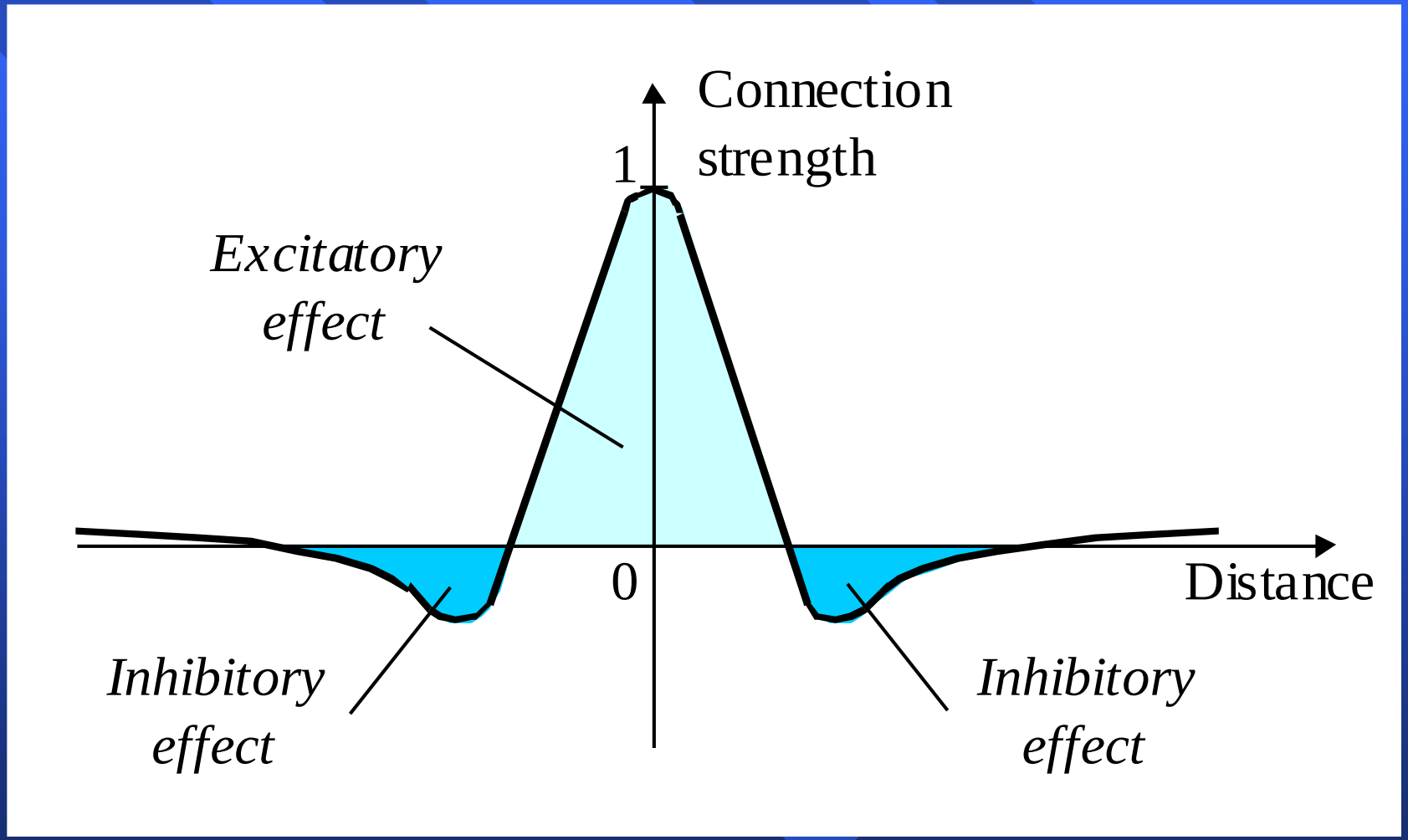
- The Kohonen model provides a topological mapping. It places a fixed number of input patterns from the input layer into a higher-dimensional output or Kohonen layer.
- Training in the Kohonen network begins with the winner's neighbourhood of a fairly large size. Then, as training proceeds, the neighbourhood size gradually decreases.

Architecture of the Kohonen Network



- The lateral connections are used to create a competition between neurons. The neuron with the largest activation level among all neurons in the output layer becomes the winner. This neuron is the only neuron that produces an output signal. The activity of all other neurons is suppressed in the competition.
- The lateral feedback connections produce excitatory or inhibitory effects, depending on the distance from the winning neuron. This is achieved by the use of a **Mexican hat function** which describes synaptic weights between neurons in the Kohonen layer.

The Mexican hat function of lateral connection



- In the Kohonen network, a neuron learns by shifting its weights from inactive connections to active ones. Only the winning neuron and its neighbourhood are allowed to learn. If a neuron does not respond to a given input pattern, then learning cannot occur in that particular neuron.
- The **competitive learning rule** defines the change Δw_{ij} applied to synaptic weight w_{ij} as

$$\Delta w_{ij} = \begin{cases} \alpha (x_i - w_{ij}), & \text{if neuron } j \text{ wins the competitio n} \\ 0, & \text{if neuron } j \text{ loses the competitio n} \end{cases}$$

where x_i is the input signal and α is the *learning rate* parameter.

- The overall effect of the competitive learning rule resides in moving the synaptic weight vector $\mathbf{W}_j$ of the winning neuron j towards the input pattern $\mathbf{X}$. The matching criterion is equivalent to the minimum **Euclidean distance** between vectors.
- The Euclidean distance between a pair of n-by-1 vectors $\mathbf{X}$ and $\mathbf{W}_j$ is defined by

$$d = \|\mathbf{X} - \mathbf{W}_j\| = \left[\sum_{i=1}^{n}(x_i - w_{ij})^2\right]^{1/2}$$

where x_i and w_{ij} are the ith elements of the vectors $\mathbf{X}$ and $\mathbf{W}_j$, respectively.

- To identify the winning neuron, $j_{\mathbf{X}}$, that best matches the input vector $\mathbf{X}$, we may apply the following condition:

$$j_{\mathbf{X}} = \min_{j} \left\| \mathbf{X} - \mathbf{W}_j \right\|, \qquad j = 1, 2, \ldots, m$$

where m is the number of neurons in the Kohonen layer.

- Suppose, for instance, that the 2-dimensional input vector **X** is presented to the three-neuron Kohonen network,

$$\mathbf{X} = \begin{bmatrix} 0.52 \\ 0.12 \end{bmatrix}$$

- The initial weight vectors, $\mathbf{W}_j$, are given by

$$\mathbf{W}_1 = \begin{bmatrix} 0.27 \\ 0.81 \end{bmatrix} \qquad \mathbf{W}_2 = \begin{bmatrix} 0.42 \\ 0.70 \end{bmatrix} \qquad \mathbf{W}_3 = \begin{bmatrix} 0.43 \\ 0.21 \end{bmatrix}$$

- We find the winning (best-matching) neuron j_x using the minimum-distance Euclidean criterion:

$$d_1 = \sqrt{(x_1 - w_{11})^2 + (x_2 - w_{21})^2} = \sqrt{(0.52 - 0.27)^2 + (0.12 - 0.81)^2} = 0.73$$

$$d_2 = \sqrt{(x_1 - w_{12})^2 + (x_2 - w_{22})^2} = \sqrt{(0.52 - 0.42)^2 + (0.12 - 0.70)^2} = 0.59$$

$$d_3 = \sqrt{(x_1 - w_{13})^2 + (x_2 - w_{23})^2} = \sqrt{(0.52 - 0.43)^2 + (0.12 - 0.21)^2} = 0.13$$

- Neuron 3 is the winner and its weight vector $\mathbf{W}_3$ is updated according to the competitive learning rule.

$$\Delta w_{13} = \alpha (x_1 - w_{13}) = 0.1(0.52 - 0.43) = 0.01$$

$$\Delta w_{23} = \alpha (x_2 - w_{23}) = 0.1(0.12 - 0.21) = -0.01$$

- The updated weight vector $\mathbf{W}_3$ at iteration $(p + 1)$ is determined as:

$$\mathbf{W}_3(p+1) = \mathbf{W}_3(p) + \Delta\mathbf{W}_3(p) = \begin{bmatrix} 0.43 \\ 0.21 \end{bmatrix} + \begin{bmatrix} 0.01 \\ -0.01 \end{bmatrix} = \begin{bmatrix} 0.44 \\ 0.20 \end{bmatrix}$$

- The weight vector $\mathbf{W}_3$ of the wining neuron 3 becomes closer to the input vector $\mathbf{X}$ with each iteration.

Competitive Learning Algorithm

Step 1: ***Initialisation.***

Set initial synaptic weights to small random values, say in an interval [0, 1], and assign a small positive value to the learning rate parameter α.

Step 2: *Activation and Similarity Matching.*

Activate the Kohonen network by applying the input vector **X**, and find the winner-takes-all (best matching) neuron j_X at iteration p, using the minimum-distance Euclidean criterion

$$j_{\mathbf{X}}(p) = \min_{j} \left\| \mathbf{X} - \mathbf{W}_j(p) \right\| = \left\{ \sum_{i=1}^{n} [x_i - w_{ij}(p)]^2 \right\}^{1/2},$$

$$j = 1, 2, \ldots, m$$

where n is the number of neurons in the input layer, and m is the number of neurons in the Kohonen layer.

Step 3: *Learning*.

Update the synaptic weights

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

where $\Delta w_{ij}(p)$ is the weight correction at iteration p.

The weight correction is determined by the competitive learning rule:

$$\Delta w_{ij}(p) = \begin{cases} \alpha[x_i - w_{ij}(p)], & j \in \Lambda_j(p) \\ 0, & j \notin \Lambda_j(p) \end{cases}$$

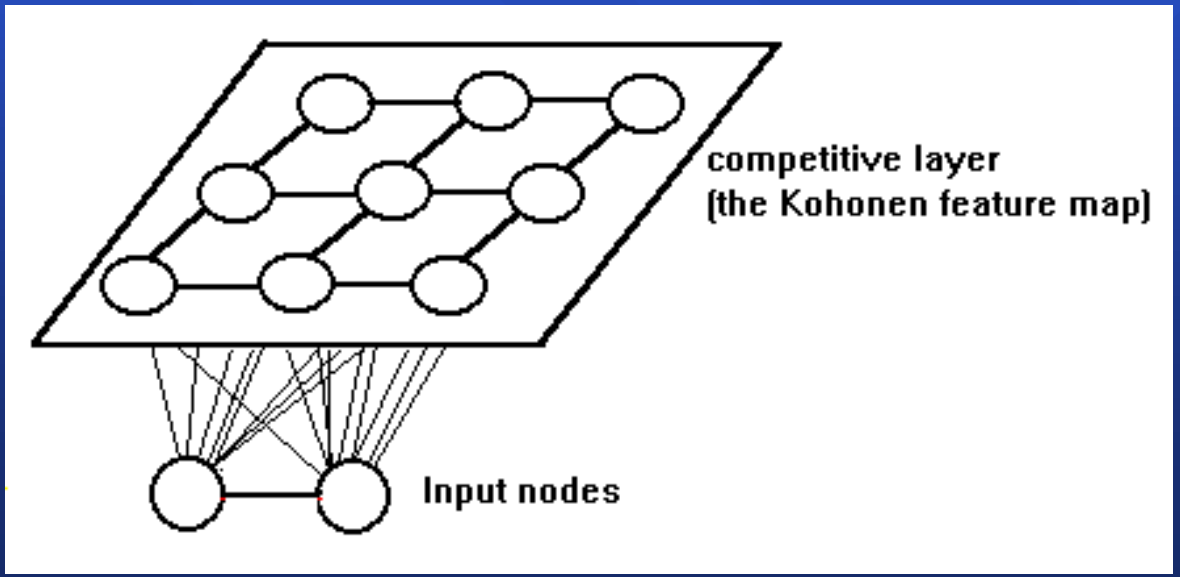
where α is the *learning rate* parameter, and $\Lambda_j(p)$ is the neighbourhood function centred around the winner-takes-all neuron j_x at iteration p.

Step 4: *Iteration.*

Increase iteration *p* by one, go back to Step 2 and continue until the minimum-distance Euclidean criterion is satisfied, or no noticeable changes occur in the feature map.

The Kohonen network (cont'd)

- The Kohonen net consists of an input layer, that distributes the inputs to every node in a second layer, known as the competitive layer.
- The competitive (output) layer is usually organised into some 2-D or 3-D surface (feature map)

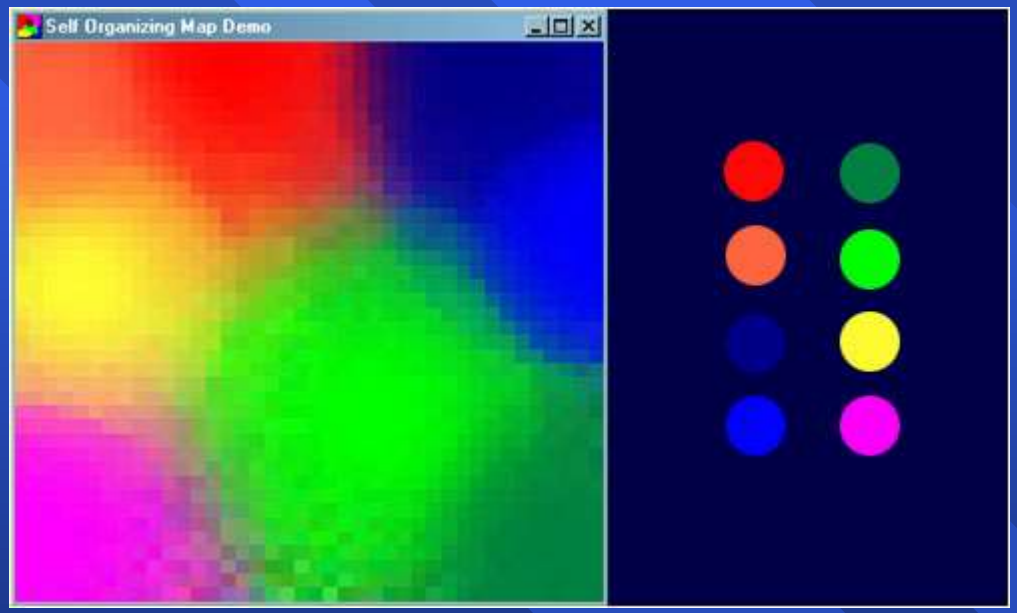


Operation of the Kohonen Net

- Each neuron in the competitive layer is connected to other neurons in its neighbourhood
- Neurons in the competitive layer have excitatory (positively weighted) connections to immediate neighbours and inhibitory (negatively weighted) connections to more distant neurons.
- As an input pattern is presented, some of the neurons in the competitive layer are sufficiently activated to produce outputs, which are fed to other neurons in their neighbourhoods
- The node with the set of input weights closest to the input pattern component values produces the largest output. This node is termed the *best matching (or winning) node*

Operation of the Kohonen Net (cont'd)

- During training, input weights of the best matching node and its neighbours are adjusted to make them resemble the input pattern even more closely
- At the completion of training, the best matching node ends up with its input weight values aligned with the input pattern and produces the strongest output whenever that particular pattern is presented
- The nodes in the winning node's neighbourhood also have their weights modified to settle down to an average representation of that pattern class
- As a result, the net is able to represent clusters of similar input patterns - a feature found useful for data mining applications, for example.



When ANNs should be applied

Difficulties with some real-life problems:

- Solutions are difficult, if not impossible, to define algorithmically due mainly to the unstructured nature
- Too many variables and/or the interactions of relevant variables not understood well
- Input data may be partially corrupt or missing, making it difficult for a logical sequence of solution steps to function effectively

When ANNs should be applied (cont'd)

- The typical ANN attempts to arrive at an answer by learning to identify the right answer through an iterative process of self-adaptation or training
- If there are many factors, with complex interactions among them, the usual "linear" statistical techniques may be inappropriate
- If sufficient data is available, an ANN can find the relevant functional relationship by means of an adaptive learning procedure from the data

Current applications of ANNs

- ANNs are good at recognition and classification tasks
- Due to their ability to recognise complex patterns, ANNs have been widely applied in character, handwritten text and signature recognition, as well as more complex images such as faces
- They have also been used successfully for speech recognition and synthesis
- ANNs are being used in an increasing number of applications where high-speed computation of functions is important, eg, in industrial robotics

Current applications of ANNs (cont'd)

- One of the more successful applications of ANNs has been as a decision support tool in the area of finance and banking
- Some examples of commercial applications of ANN are:
  - Financial market analysis for investment decision making
  - Sales support - targeting customers for telemarketing
  - Bankruptcy prediction
  - Intelligent flexible manufacturing systems
  - Stock market prediction
  - Resource allocation – scheduling and management of personnel and equipment

ANN applications - broad categories

- According to a survey (Quaddus & Khan, 2002) covering the period 1988 up to mid 1998, the main business application areas of ANNs are:
  - Production (36%)
  - Information systems (20%)
  - Finance (18%)
  - Marketing & distribution (14.5%)
  - Accounting/Auditing (5%)
  - Others (6.5%)

ANN applications - broad categories (cont'd)

Table 1: Distribution of the Articles by Areas and Year

AREA	1988	89	90	91	92	93	94	95	96	97	98	Total	% of Total
Accounting/Auditing	1	0	1	1	6	3	3	7	7	5	0	34	4.97
Finance	0	0	4	11	19	28	27	18	5	9	2	123	17.98
Human resources	0	0	0	1	0	1	1	0	0	0	0	3	0.44
Information system	4	6	9	7	15	24	21	18	13	18	3	138	20.18
Marketing/Distributi	2	2	2	3	8	10	12	17	29	14	0	99	14.47
Production	2	6	8	21	31	38	24	50	29	31	1	241	35.23
Others	0	0	1	7	3	8	7	8	7	5	0	46	6.73
Yearly Total	**9**	**14**	**25**	**51**	**82**	**112**	**95**	**118**	**90**	**82**	**6**	**684**	**100.00**
% of Total	**1.32**	**2.05**	**3.65**	**7.46**	**11.99**	**16.37**	**13.89**	**17.25**	**13.16**	**11.99**	**0.88**	**100.00**	

- The levelling off of publications on ANN applications may be attributed to the ANN moving from the research to the commercial application domain
- The emergence of other intelligent system tools may be another factor