Machine Learning (part II)

Kohonen Map

Angelo Ciaramella

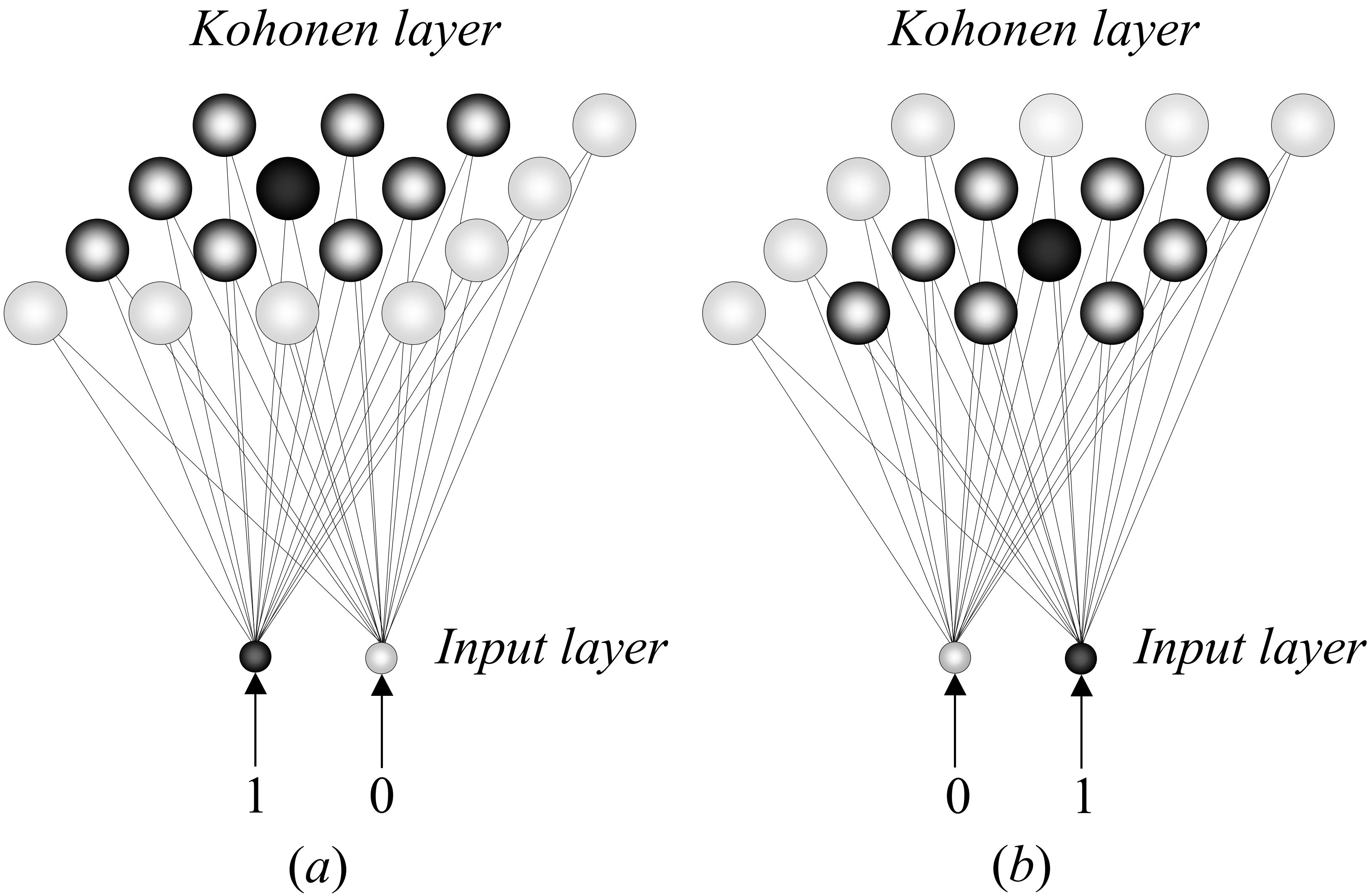
Competitive learning

- competitive learning
  - neurons compete among themselves to be activated
  - only a single output neuron is active at any time
  - 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

Self Organizing Map

- Human brain
  - Dominated by the cerebral cortex
  - 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, somatosensory, etc.), and associated with different sensory inputs.
  - 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

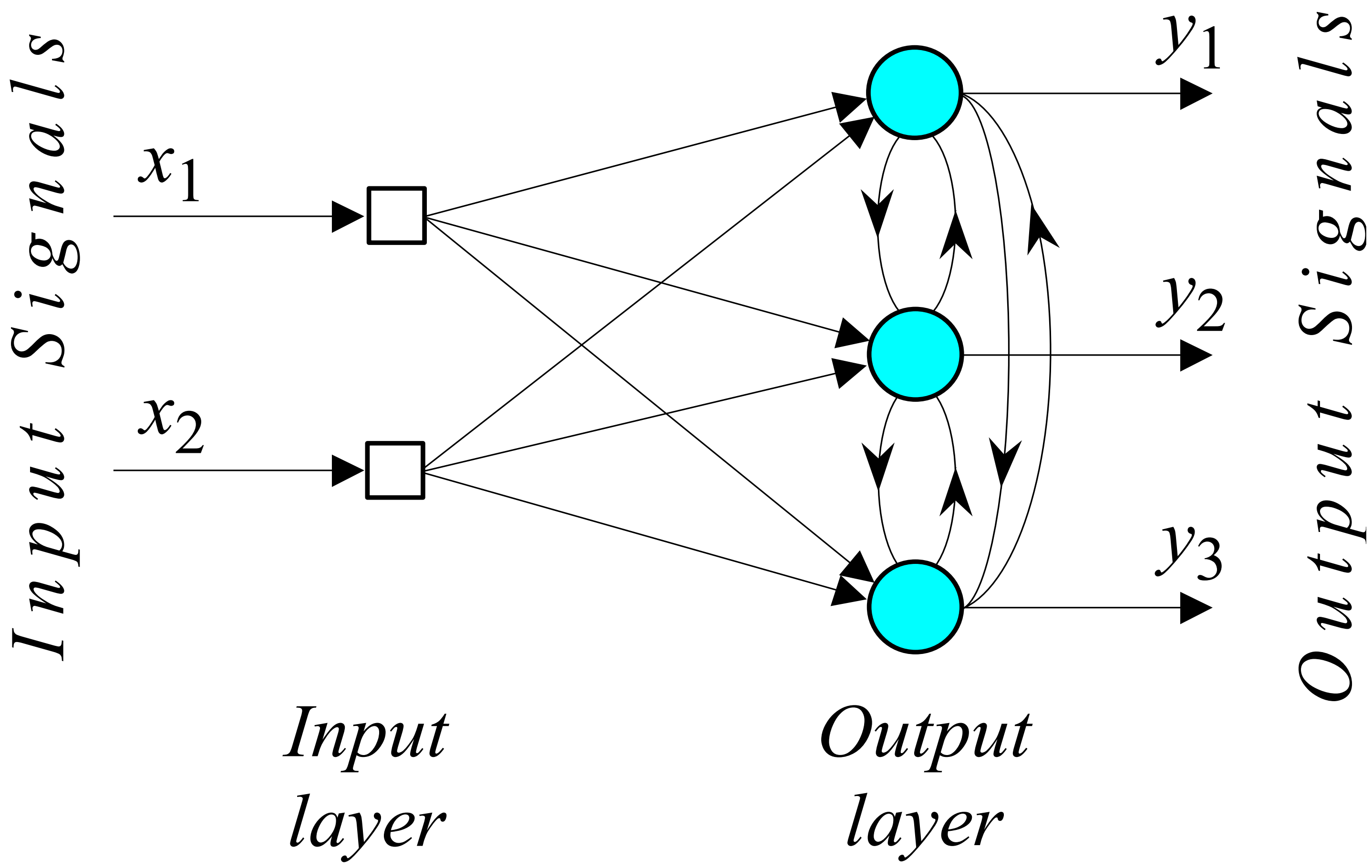
Self Organizing Map



The Kohonen Net

- Kohonen Neural Network
  - provides a topological mapping (topographic map)
  - 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
  - as training proceeds the neighbourhood size gradually decreases

Architecture



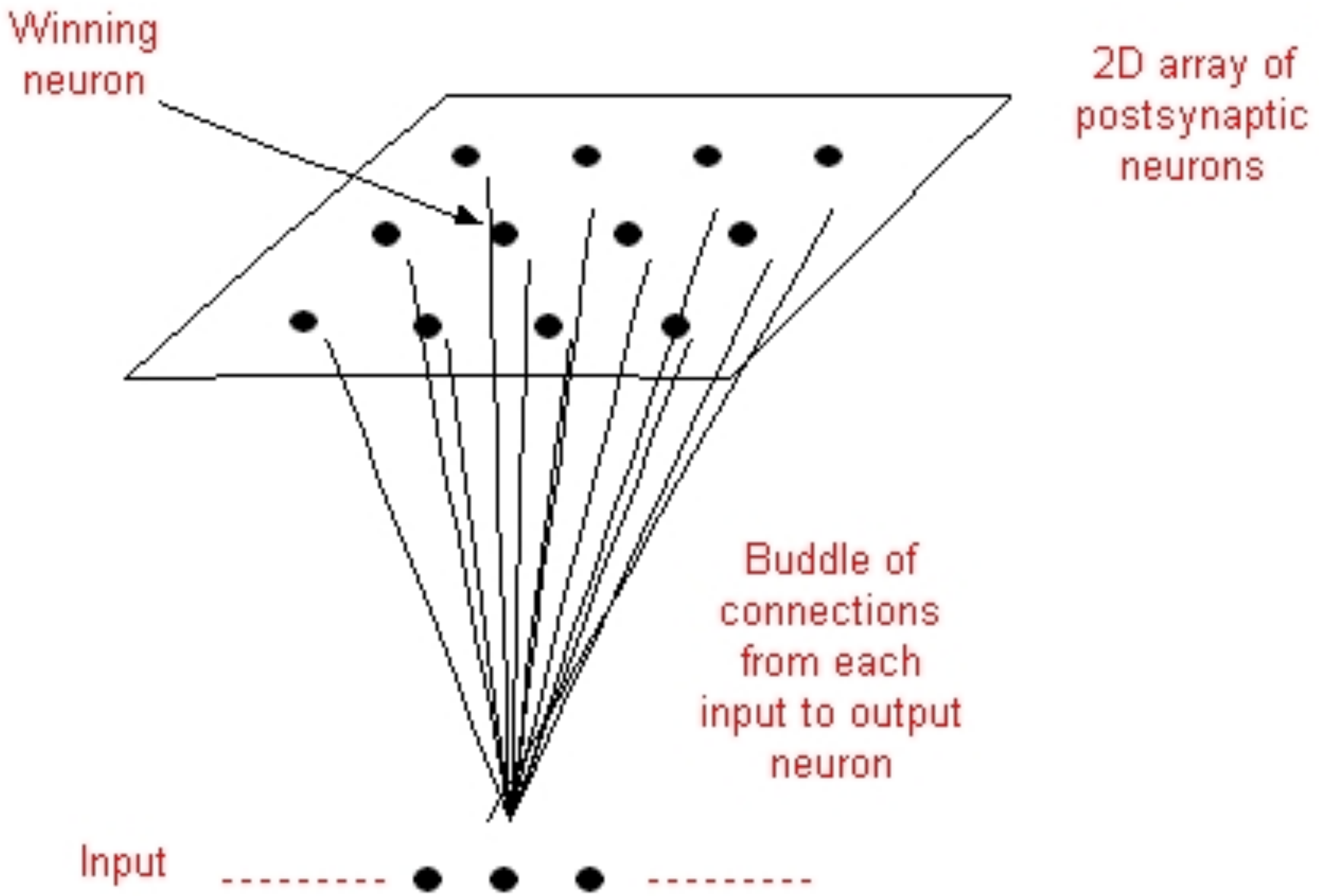
Connections

- The lateral connections are used to create a competition between neurons.

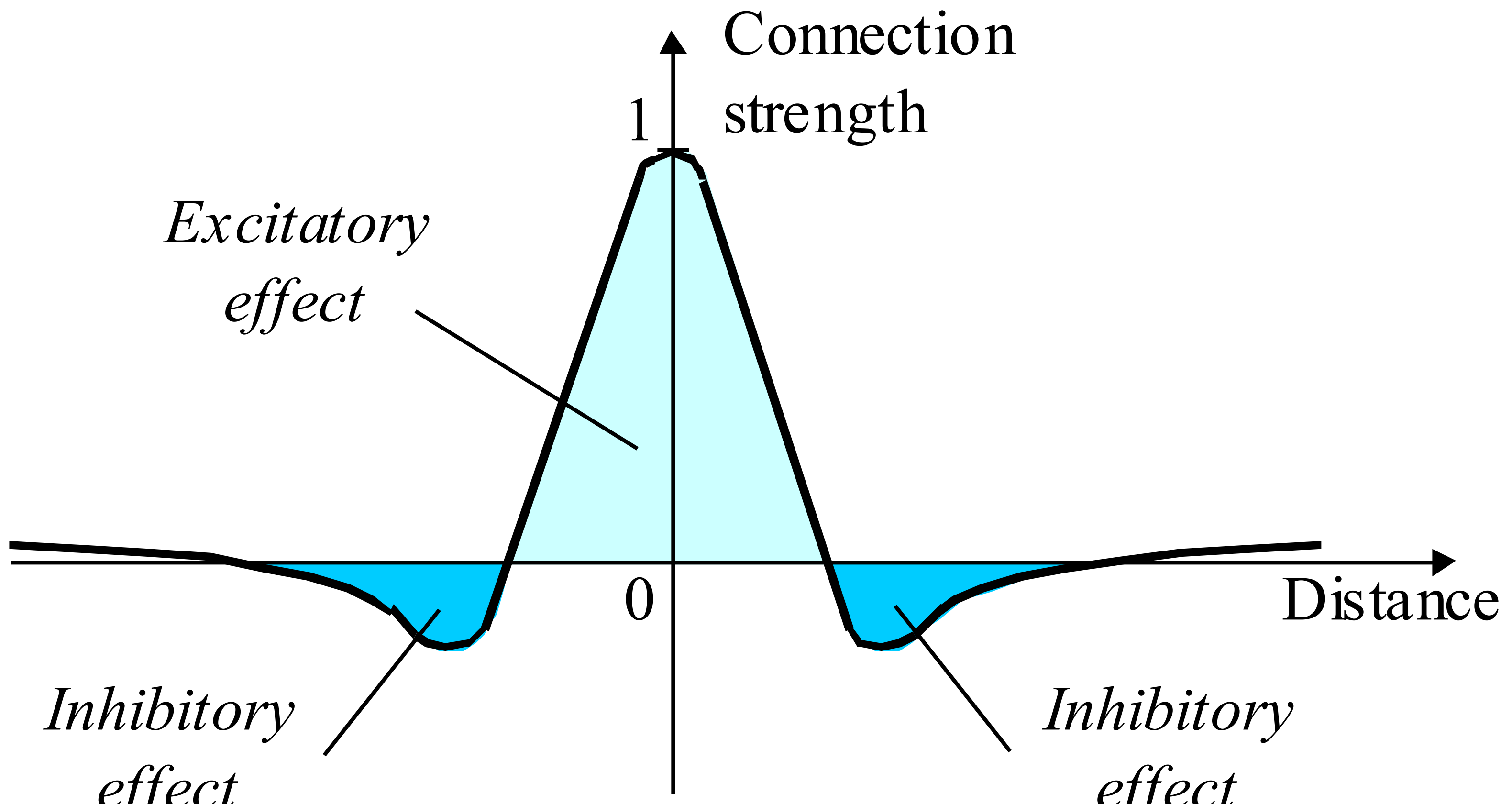
- The neuron with the largest activation level among all neurons in the output layer becomes the winner
  - neuron that produces an output signal
  - the activity of all other neurons is suppressed in the competition

- lateral feedback connections produce
  - excitatory or inhibitory effects, depending on the distance from the winning neuron
  - Mexican hat function which describes synaptic weights between neurons in the Kohonen layer

Winning neuron



Mexican hat function



Learning rule

- Competitive learning rule

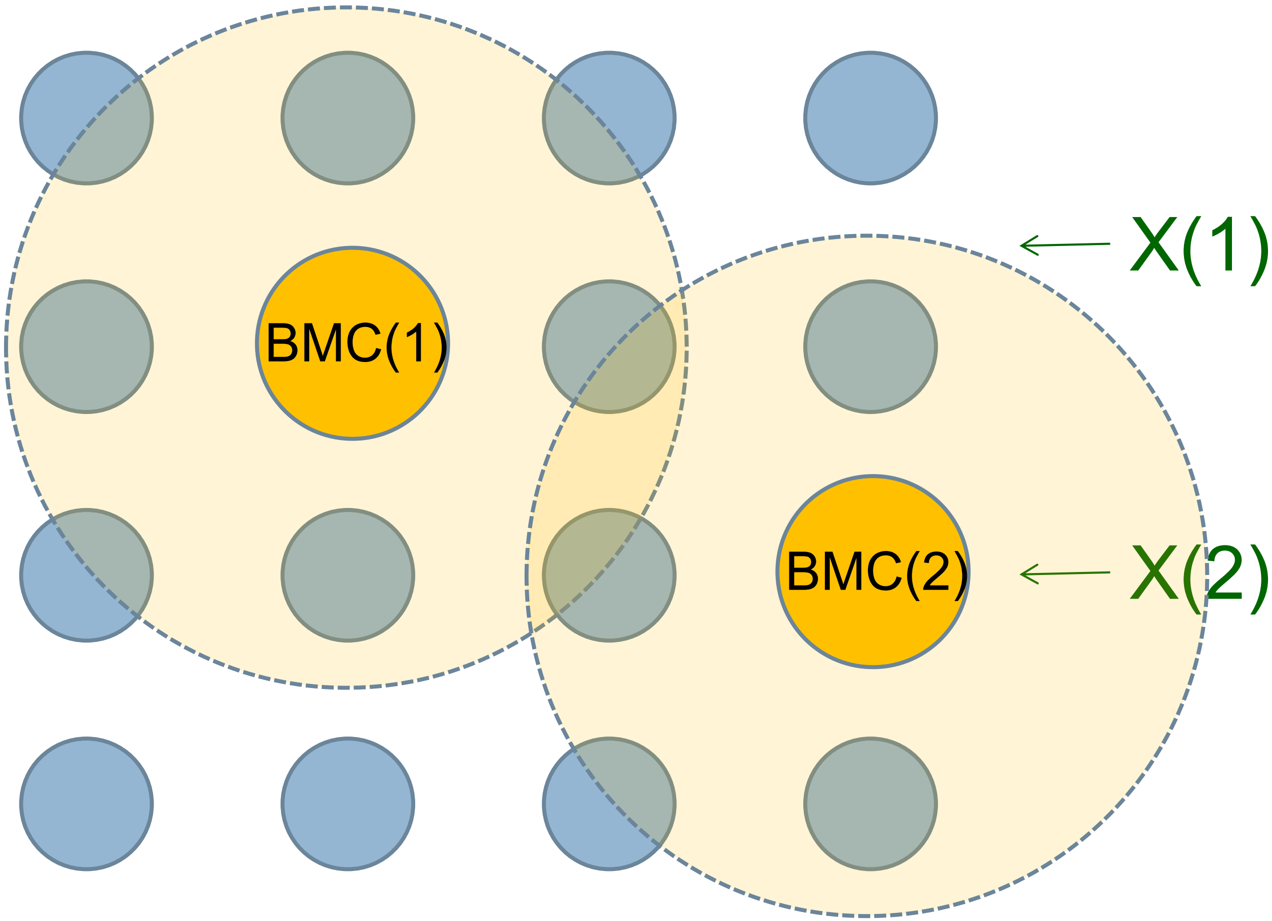
$$\Delta w_{ij} = \begin{cases} \alpha\,(x_i - w_{ij}), & \text{if neuron } j \text{ wins the competition} \\ 0, & \text{if neuron } j \text{ loses the competition} \end{cases}$$

- The winning neuron (Best Matching Unit)

$$j_{\mathbf{X}} = \min_{j} \left\| \mathbf{X} - \mathbf{W}_j \right\|,$$

Best Meatching Unit

neurons
(m_i)



Example

- Learning example

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

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

Example

- weight vector W_3 of the wining neuron 3 becomes closer to the input vector X with each iteration

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

Algorithm

- Step 1: Initialization
  - 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

- Step 2: Activation and Similarity Matching

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

Algorithm

- Step 3: learning

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

neighbourhood function

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

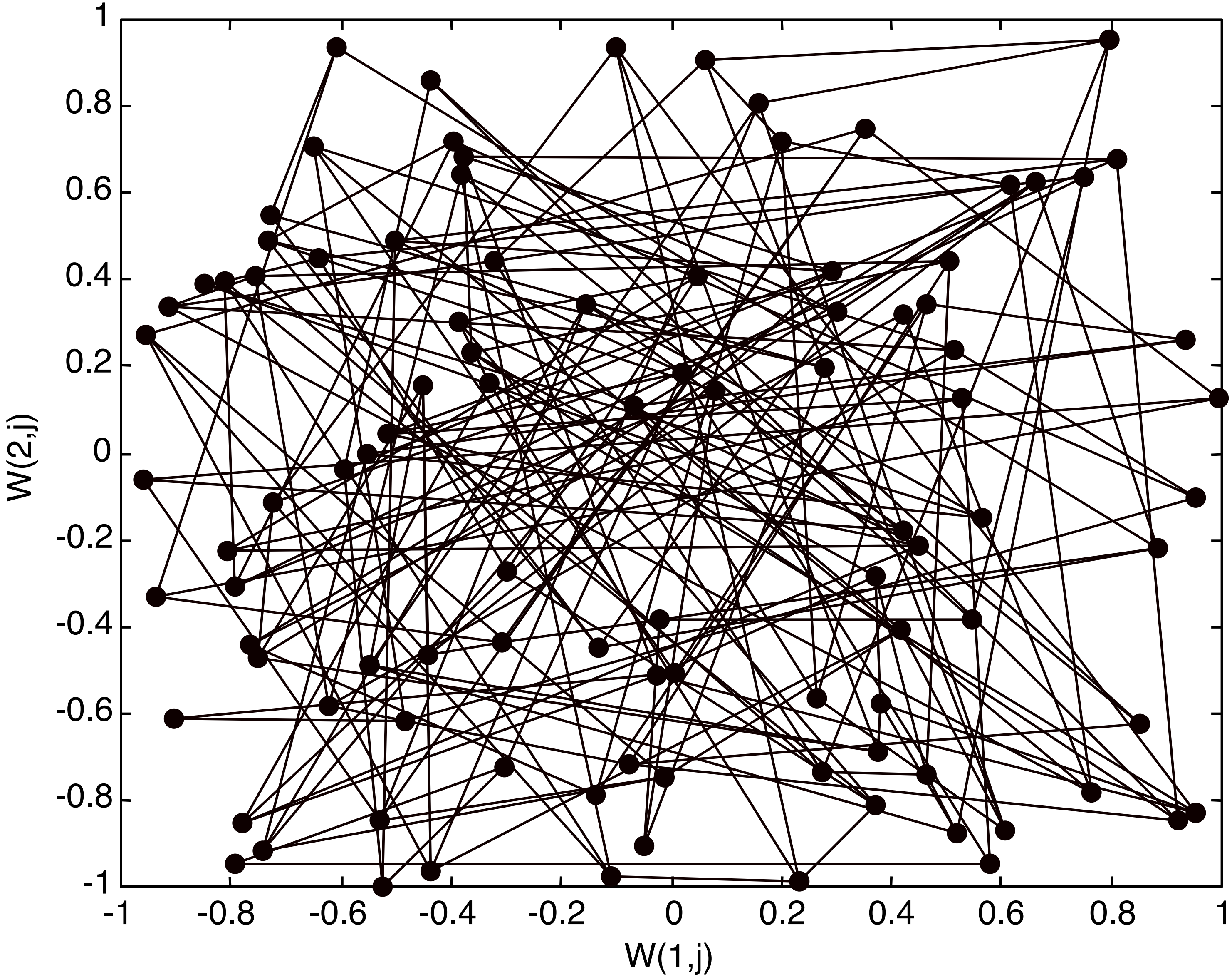
Algorithm

- 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

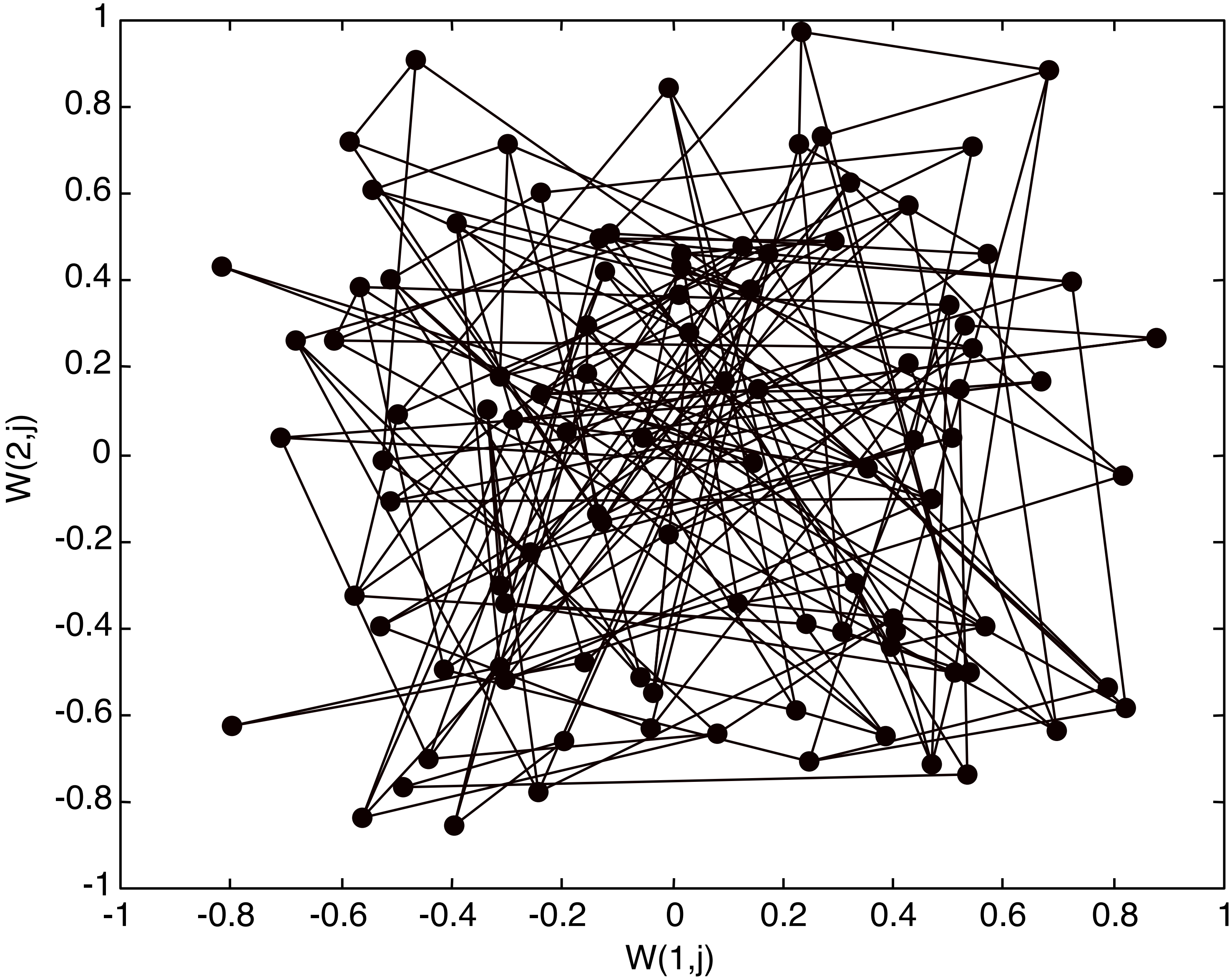
Learning example

- consider the Kohonen network with 100 neurons arranged in the form of a two-dimensional lattice with 10 rows and 10 columns
- The network is required to classify two-dimensional input vectors
- The network is trained with 1000 two-dimensional input vectors generated randomly in a square region in the interval between –1 and +1
- The learning rate parameter is equal to 0.1

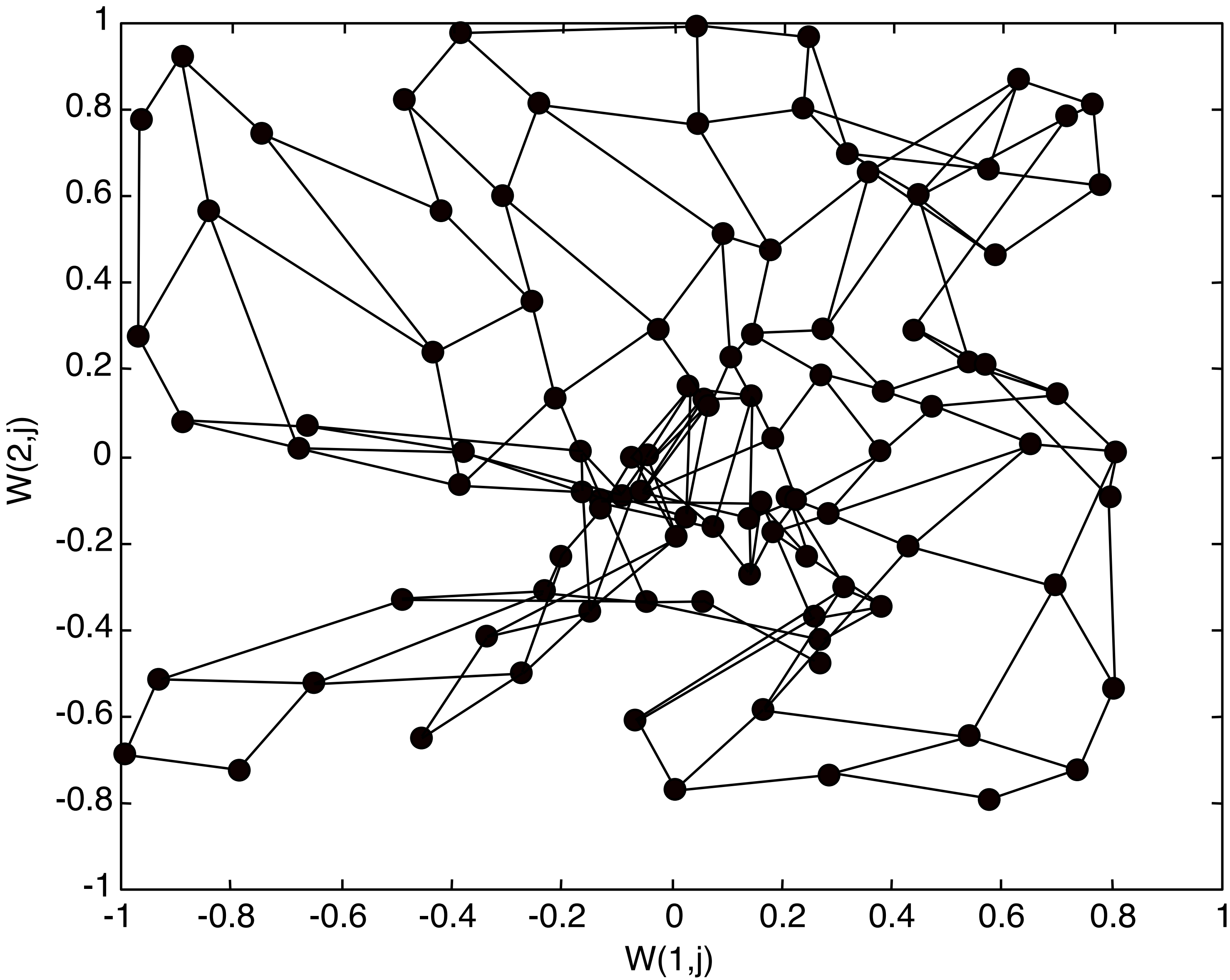
Initial random weights



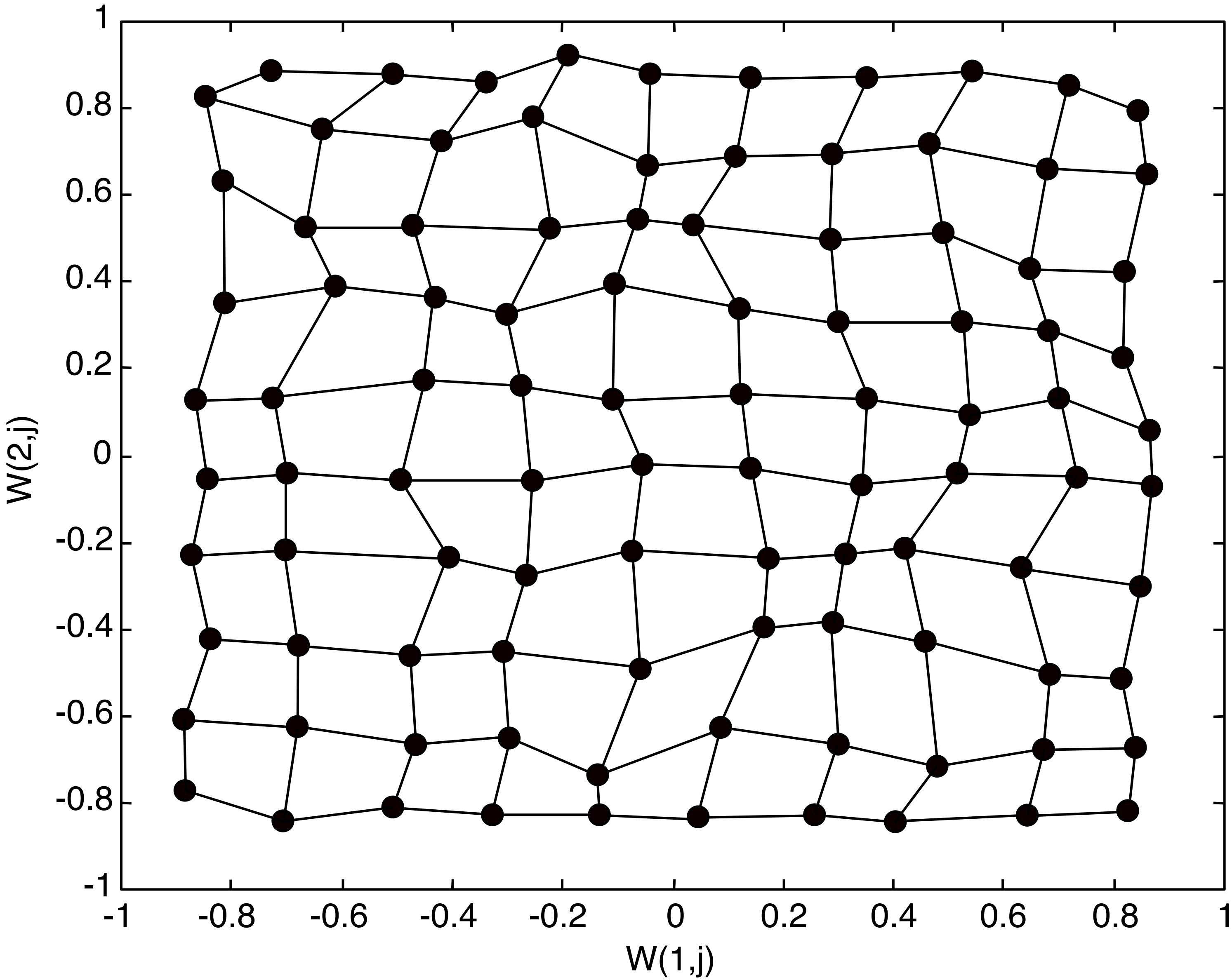
Net after 100 iterations



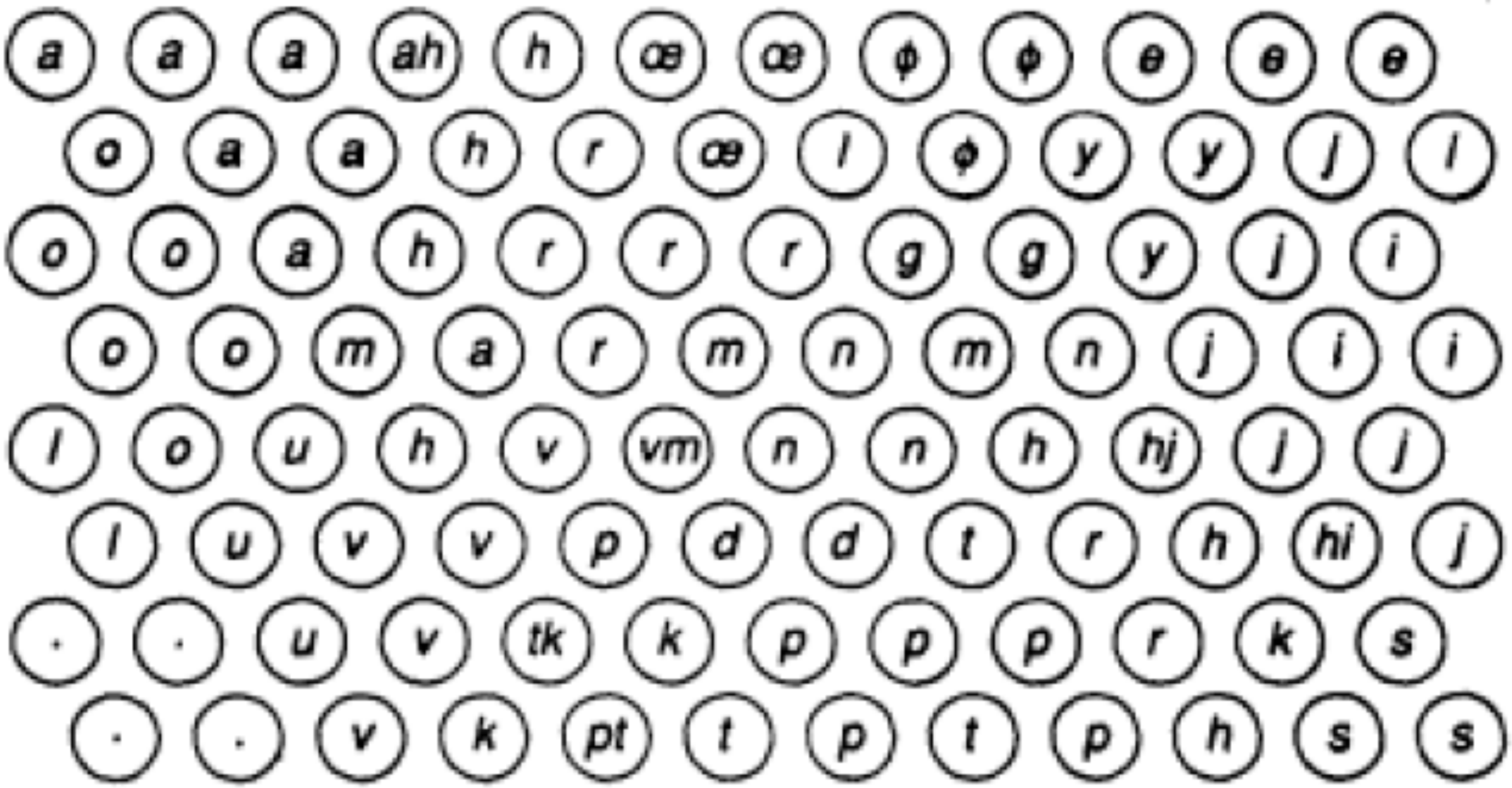
Net after 1000 iterations



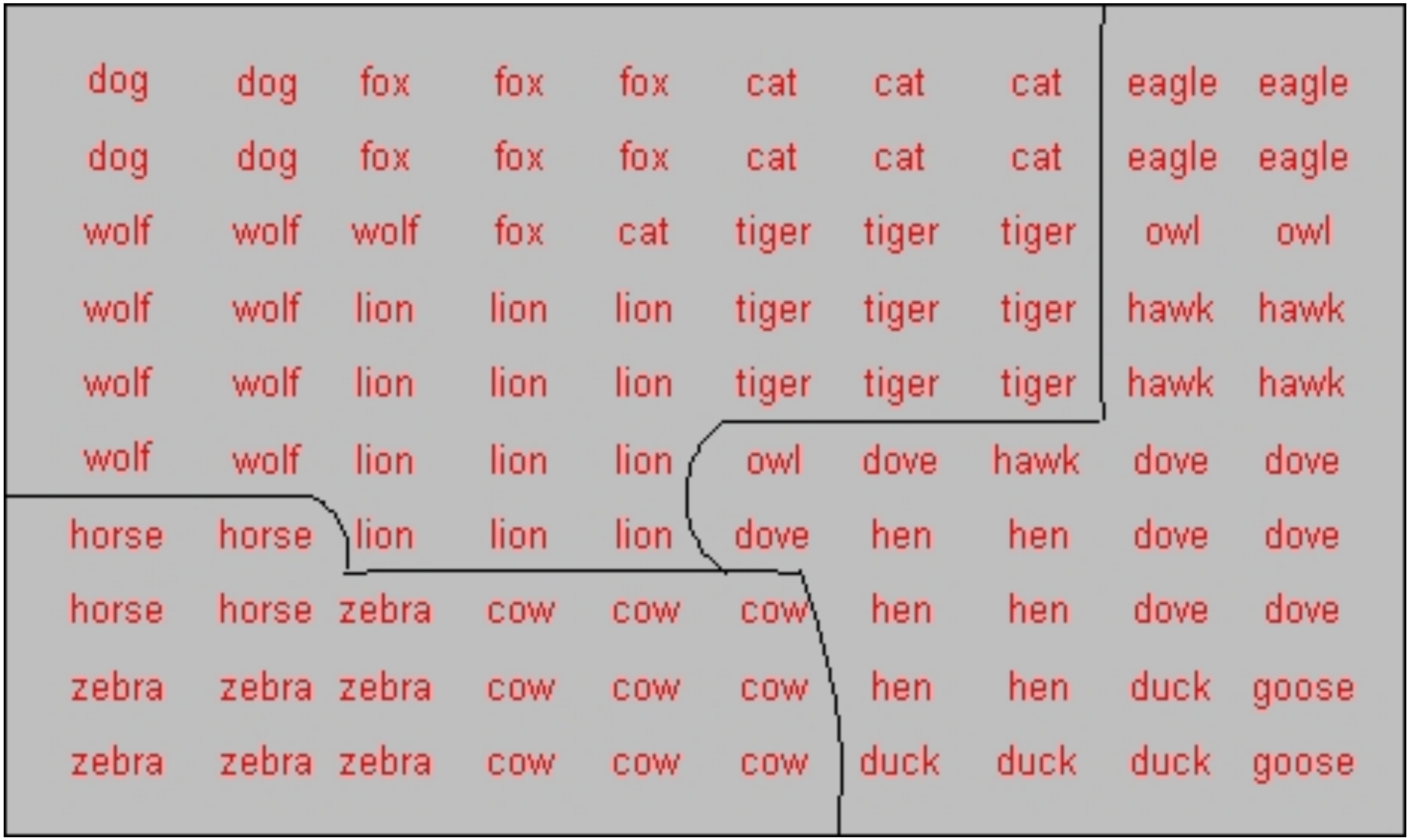
Net after 10.000 iterations



Presenting SOM

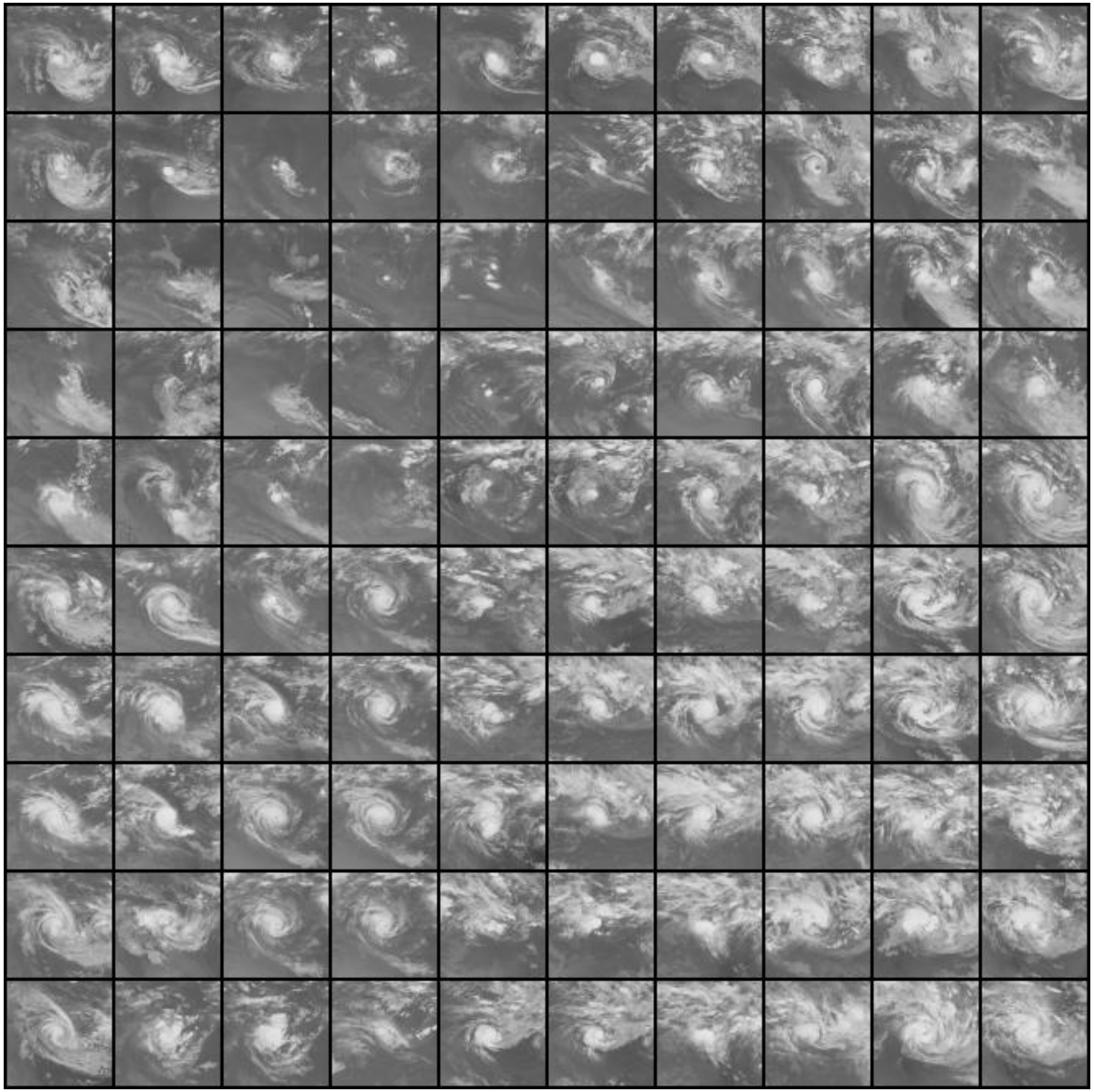


Examples


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

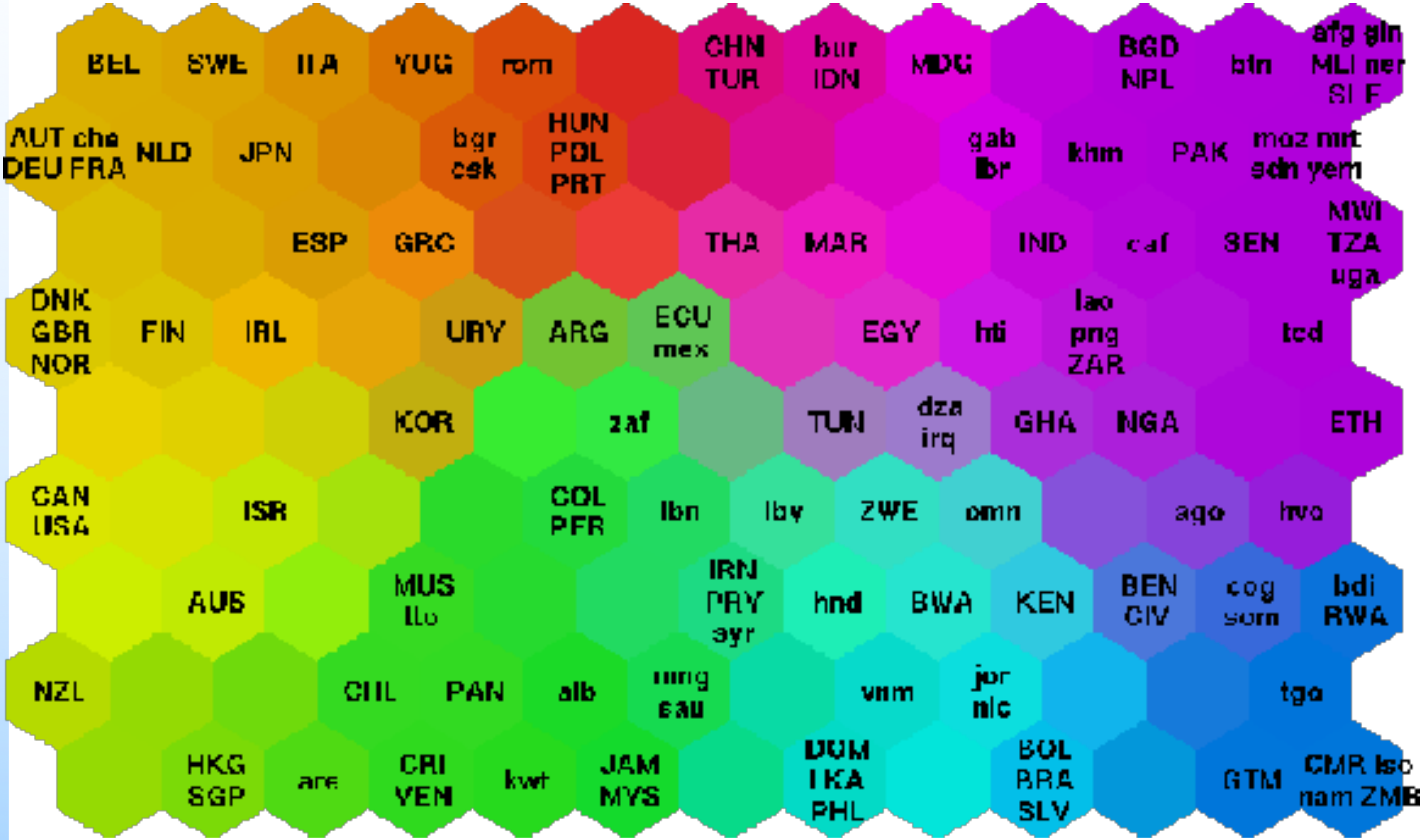
Examples

Examples



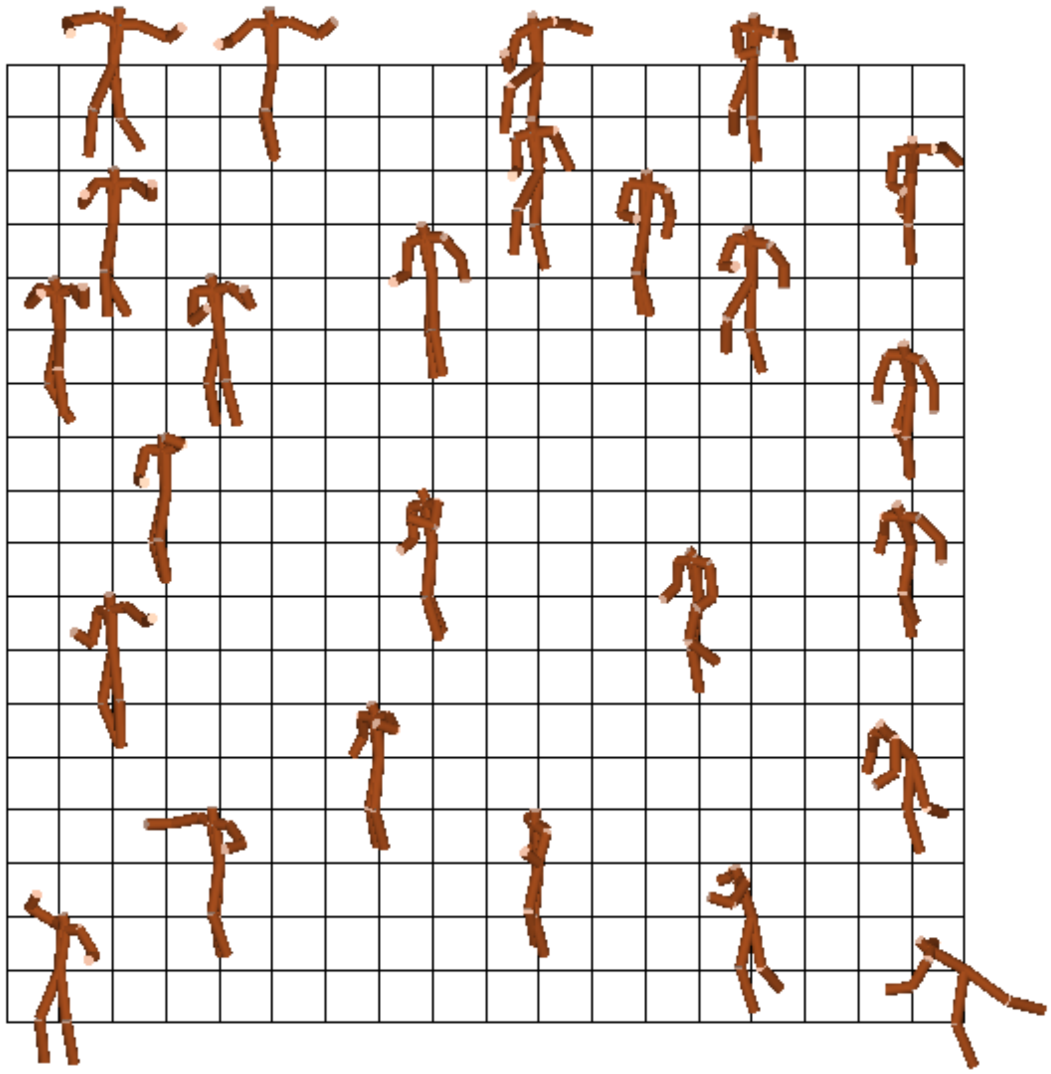
ML – Kohonen Map

Examples



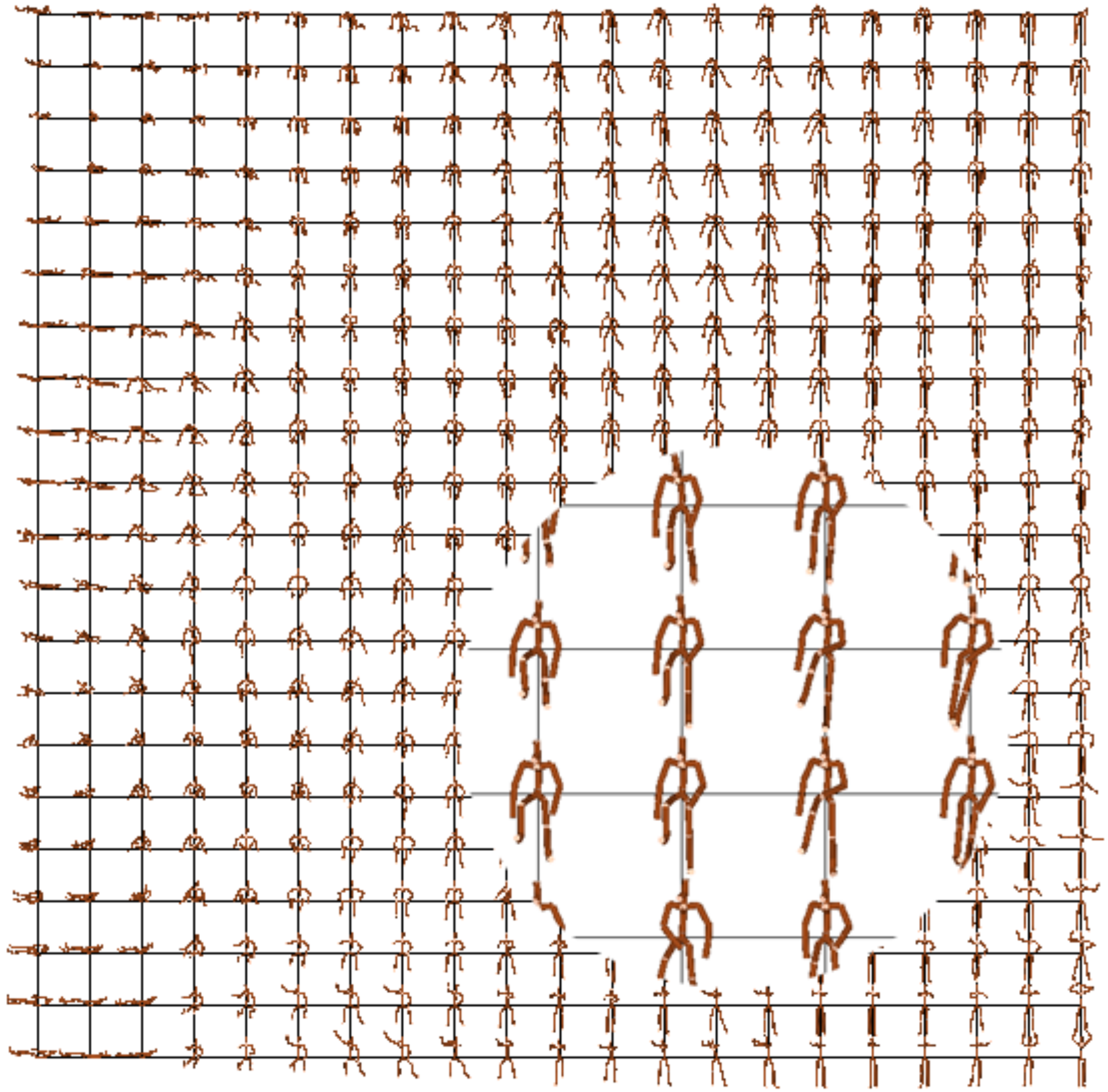
ML – Kohonen Map

Examples



16 dance styles

Examples



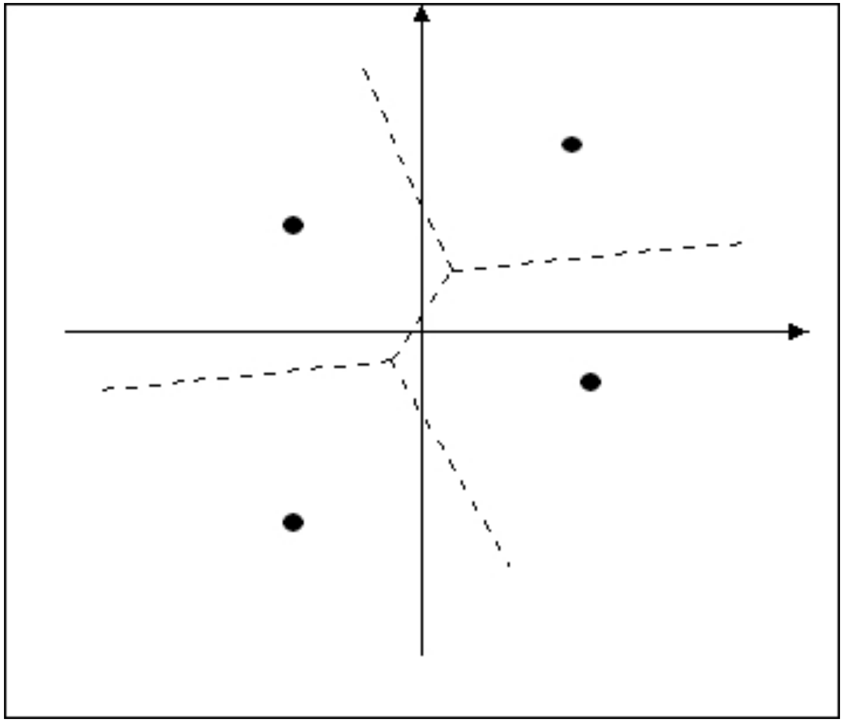
436 posture samples

Vector Quantization

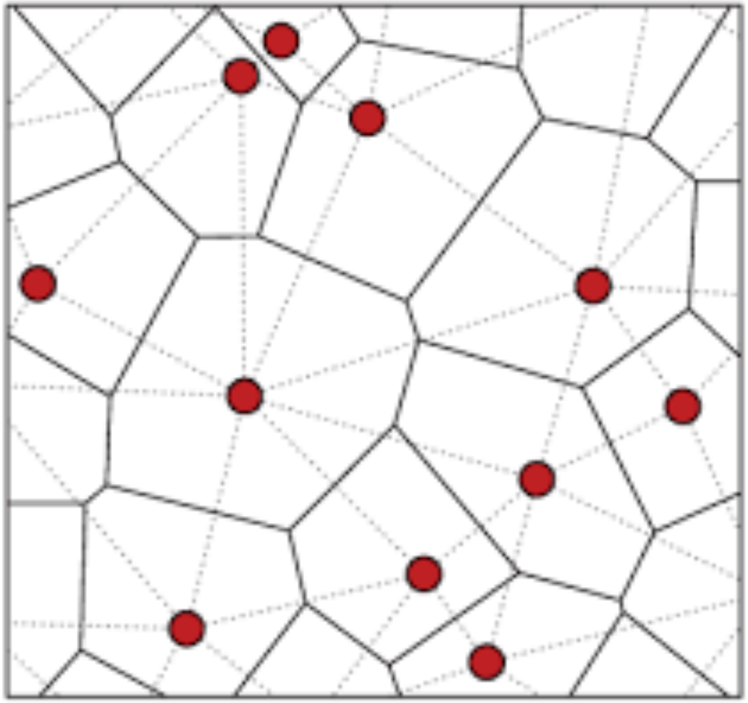
- Vector Quantisation
  - technique that exploits the underlying structure of input vectors for the purpose of data compression
  - input space is divided in a number of distinct regions and for each region a reconstruction (representative) is defined
  - new input vector
    - the region in which the vector lies is first determined
    - represented by the reproduction vector for this region
  - The collection of all possible reproduction vectors is called the code book of the quantizer and its members are called code words

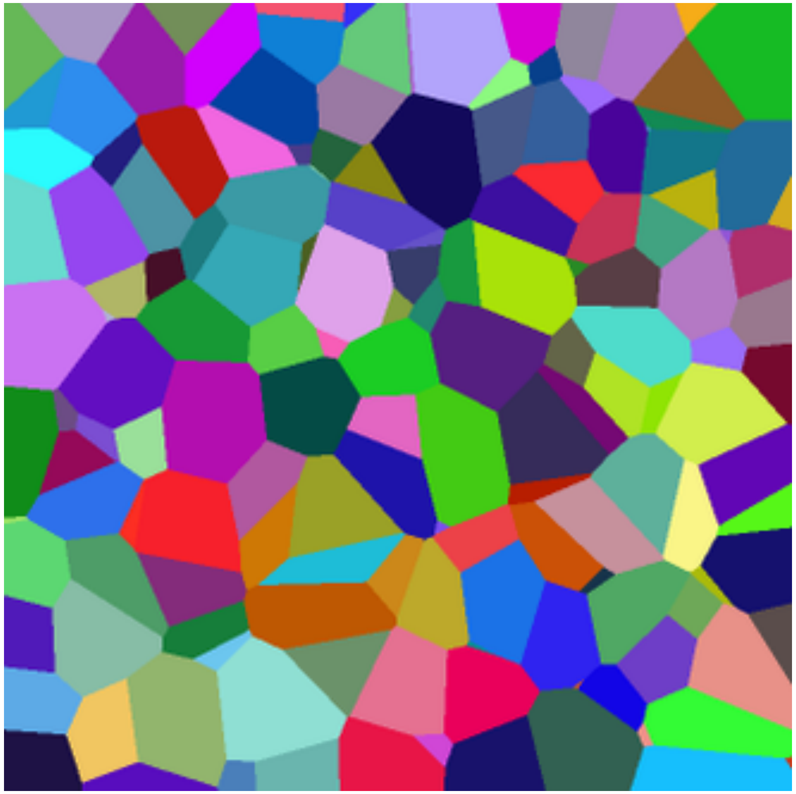
Voronoi tassellation

- A vector quantizer with minimum encoding distortion
  - Voronoi tassellattion or nearest-neighbour quantizer
  - partition of the space according to the nearest-neighbour rule based on the Euclidean metric



Voronoi tassellation

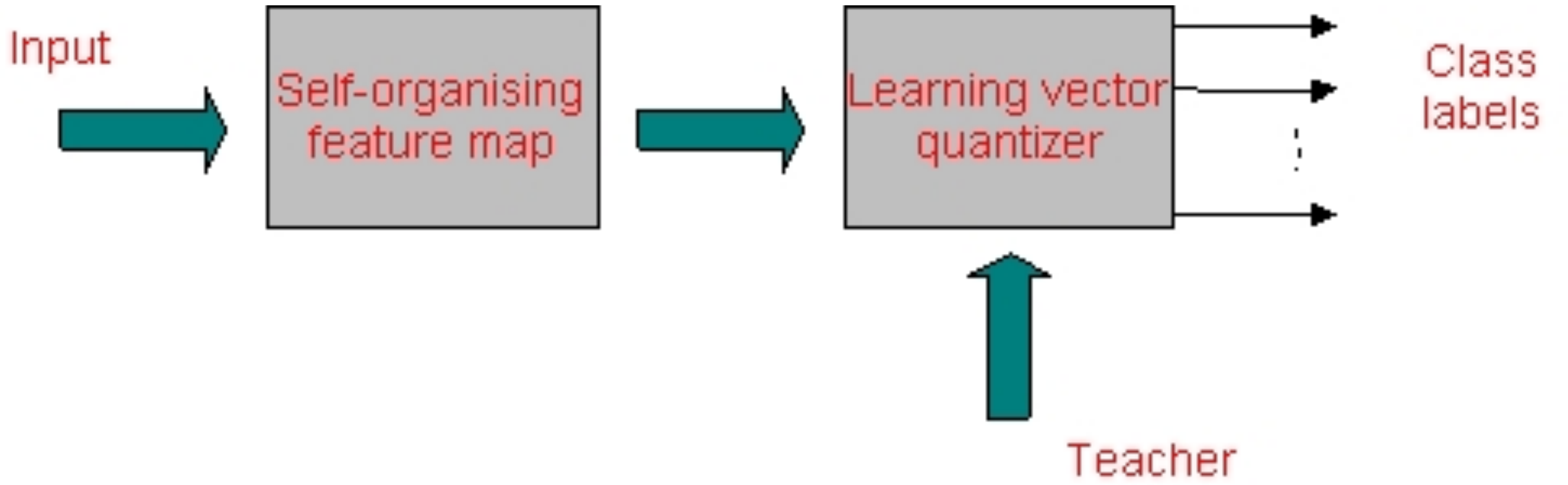




Learning Vector Quantization

- SOM
  - provides an approximate method for computing the Voronoi vectors in an unsupervised manner
- Learning Vector Quantisation
  - supervised learning technique that uses class information to move the Voronoi vectors slightly, so as to improve the quality of the classifier decision regions

Learning Vector Quantization



Learning Vector Quantization

- LVQ algorithm

$$\mathbf{w}_c(n+1) = \mathbf{w}_c(n) + \alpha(\mathbf{x}_i - \mathbf{w}_c(n)) \qquad C_{x_i} = C_{w_c}$$

$$\mathbf{w}_c(n+1) = \mathbf{w}_c(n) - \alpha(\mathbf{x}_i - \mathbf{w}_c(n)) \qquad C_{x_i} \neq C_{w_c}$$