Artificial Neural Networks: Self Organizing Networks

Artificial neural networks which are currently used in tasks such as speech and handwriting recognition are based on learning mechanisms in the brain i.e synaptic changes. In addition, one kind of artificial neural network, self organizing networks, is based on the topographical organization of the brain.

The somatosensory map - homunculus

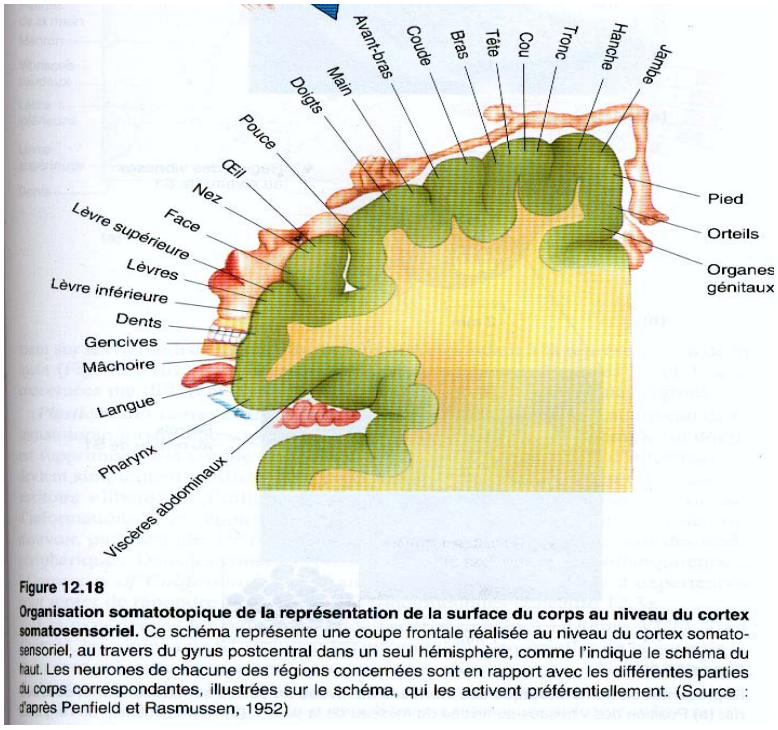


Figure 12.18

Organisation somatotopique de la représentation de la surface du corps au niveau du cortex somatosensoriel. Ce schéma représente une coupe frontale réalisée au niveau du cortex somatosensoriel, au travers du gyrus postcentral dans un seul hémisphère, comme l'indique le schéma du haut. Les neurones de chacune des régions concernées sont en rapport avec les différentes parties du corps correspondantes, illustrées sur le schéma, qui les activent préférentiellement. (Source : d'après Penfield et Rasmussen, 1952)

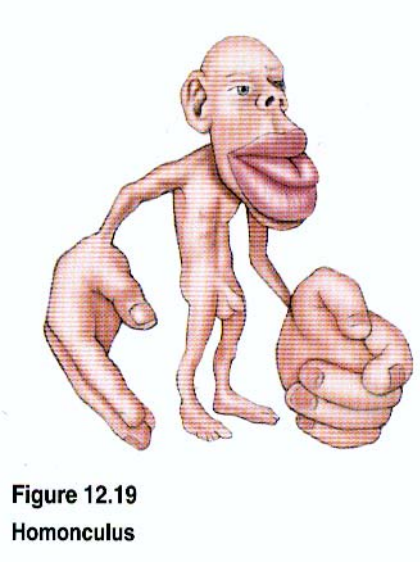


Figure 12.19

Homonculus

The somatosensory cortex is organized in a topographic fashion. i.e. the representation of the body in the somatosensory cortex reflects the topographic arrangement of the body.

The cortical map - homunculus

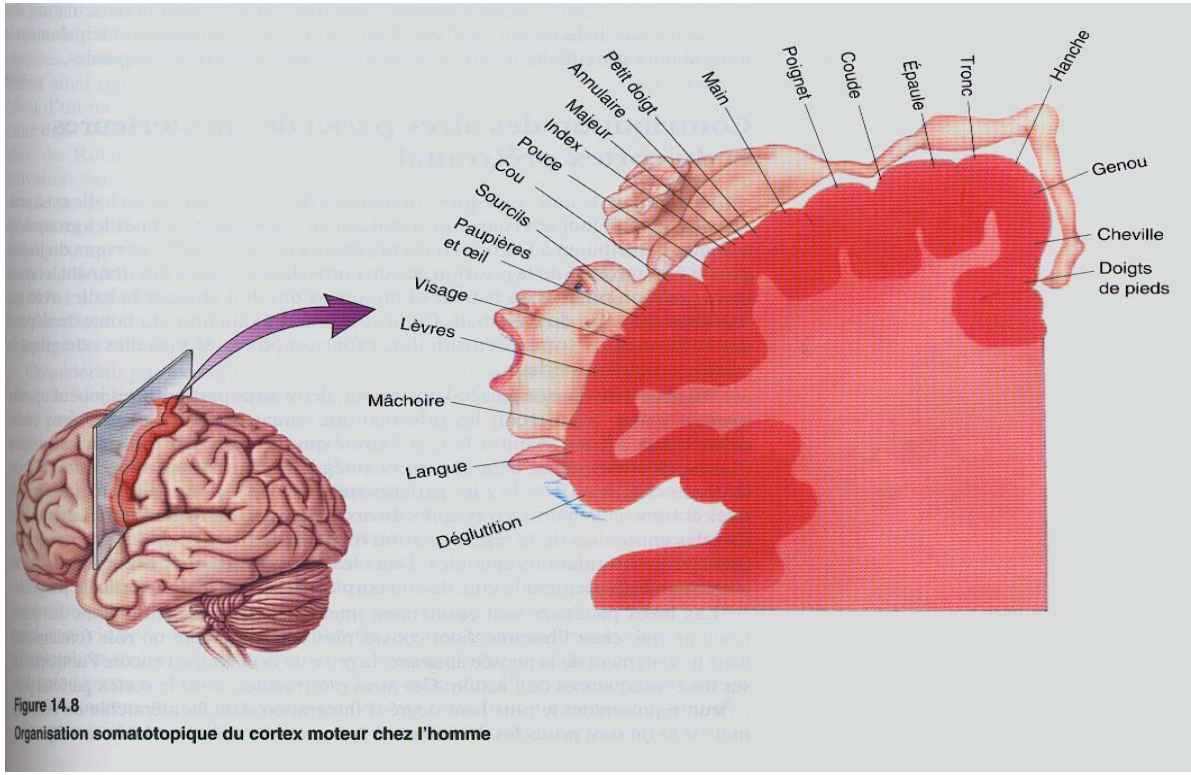
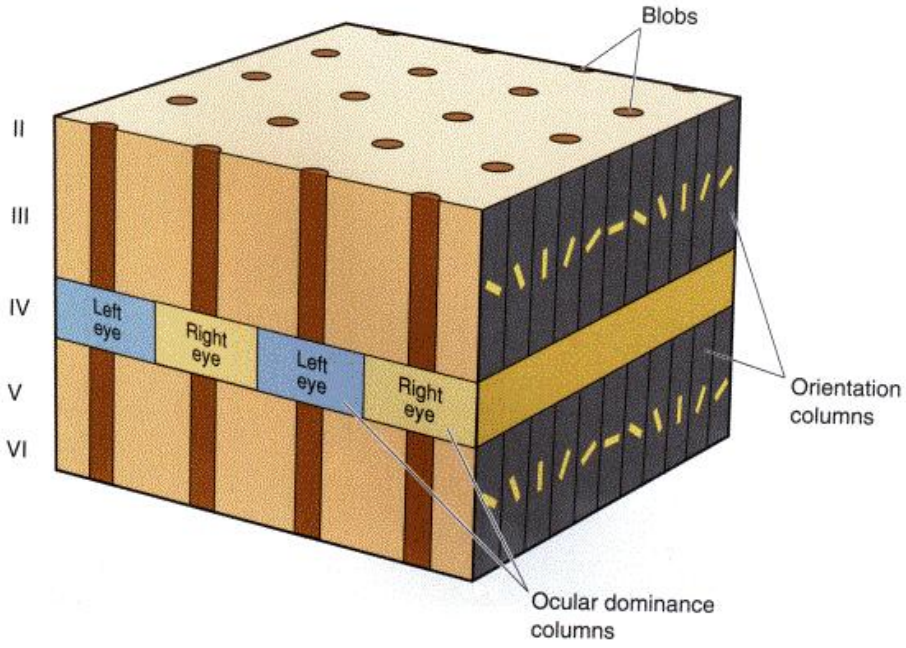


Figure 14.8
Organisation somatotopique du cortex moteur chez l'homme

The motor cortex is organized in a topographic fashion. i.e. the representation of the body in the motor cortex reflects the topographic arrangement of the body.

The visual orientation map



Neurons in the visual cortex respond to bars of light.
Neurons that respond to similar bars are located next to each other in the visual cortex.

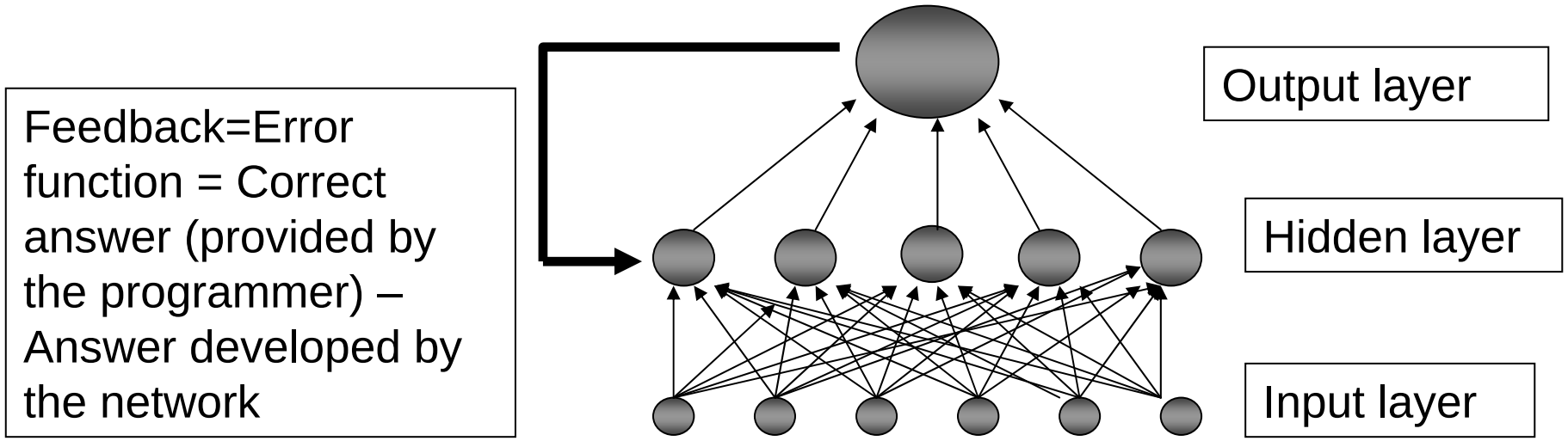
Self organizing maps are based on these principles:

- Learning by changes in synaptic weight
- A topographic organization of information ie Similar information is found in a similar spatial location.

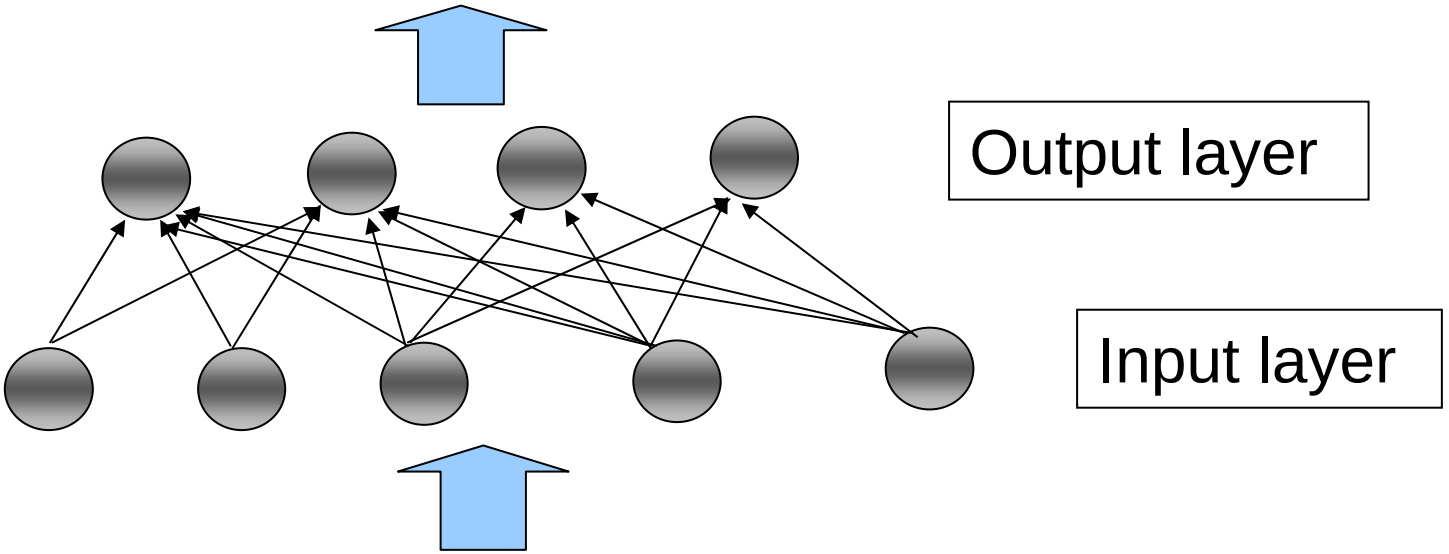
Self organizing maps are a type of artificial neural network. Unlike methods like back propagation, self organizing networks are unsupervised, hence the name self organizing.

Backpropagation requires supervision

Supervision – The correct answer is represented in the network by the investigator.



No error feedback in the unsupervised network



Example task – classification of 4 vectors into 2 groups. Simple task but it will help us to identify the principles involved. We will then later move to a more complex example

Vectors to be classified

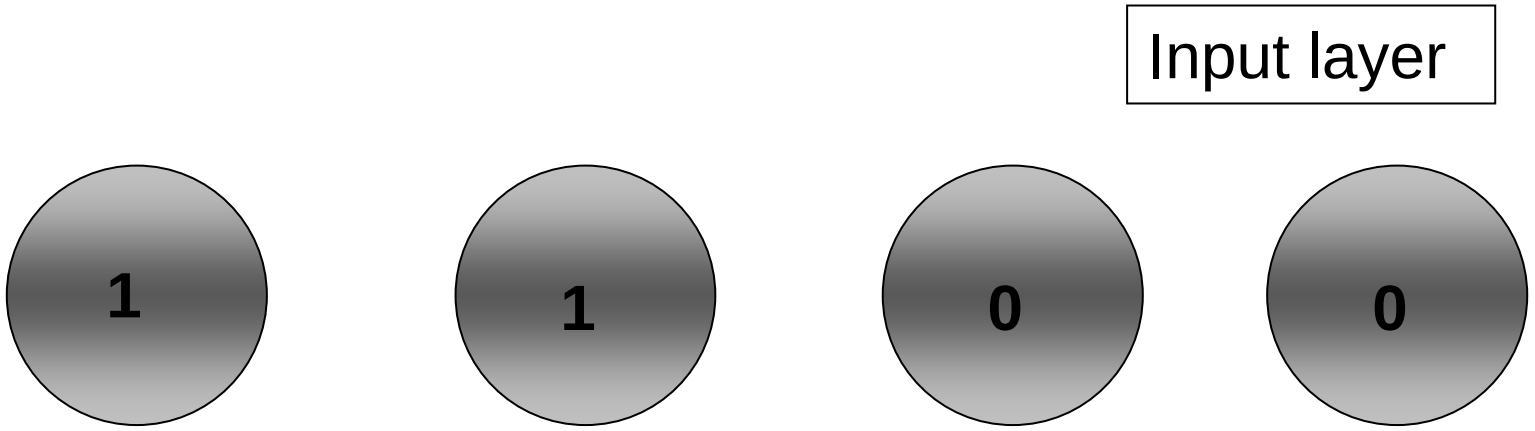
(1,1,0,0)

(0,0,0,1)

(1,0,0,0)

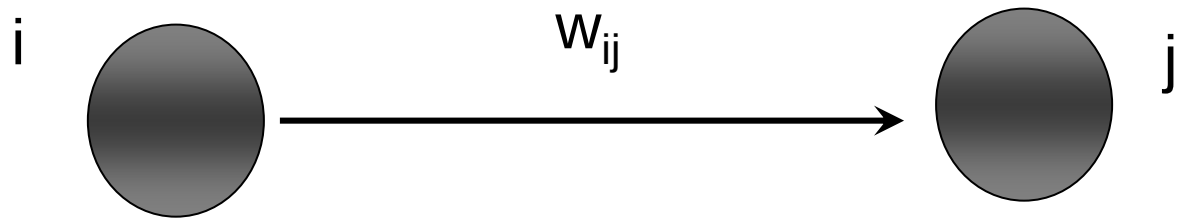
(0,0,1,1)

The input layer



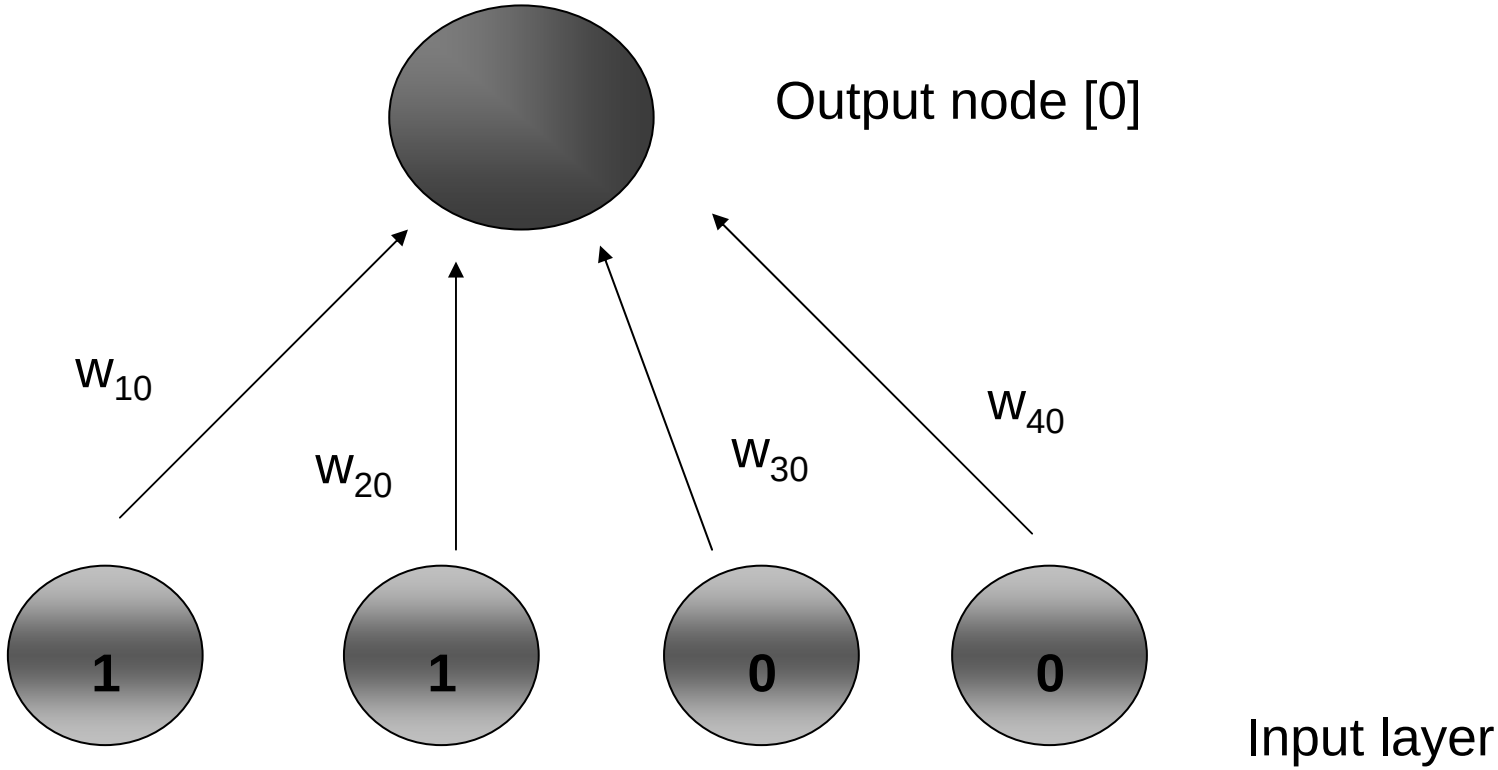
The input layer is also called an input vector. Each element or node in the input layer is associated with a value. In our case we will call it x_i. The node can also be loosely identified with a neuron or an assembly of neurons.

The output layer is also made up of several units called nodes. These nodes can be loosely identified as neurons or assemblies of neurons



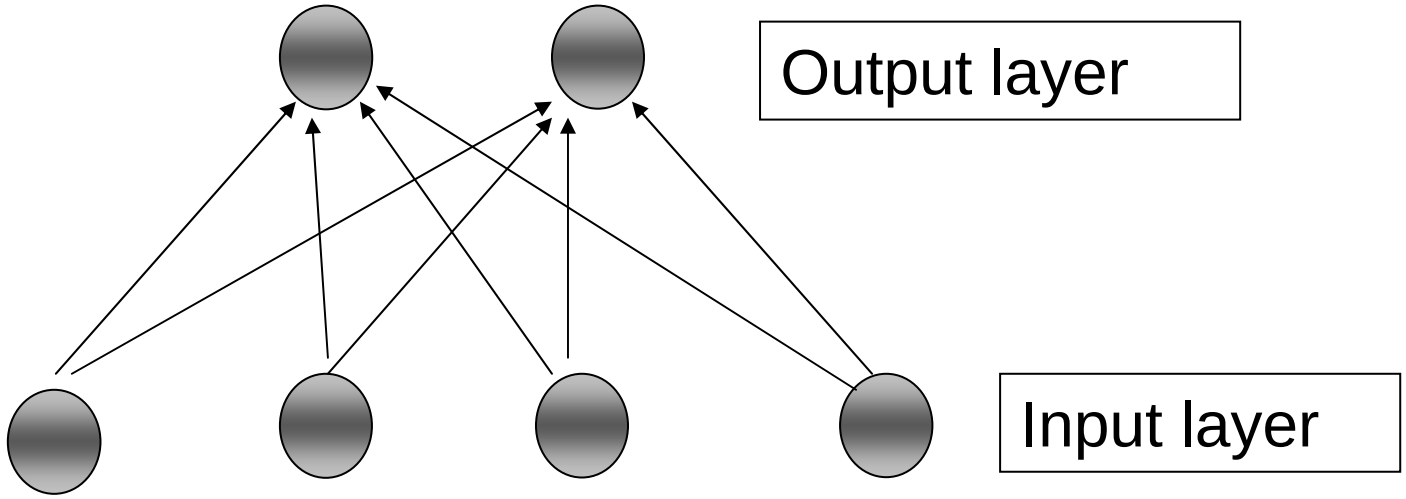
The nodes influence each other through connections w_{ij} .This connection can be identified with a synapse. So the node i can be thought of as the presynaptic neuron and node j as the postsynaptic neuron.

Connectivity



Every element in the input layer is connected to all the nodes in the output layer. The weight vector of the output node will have as many elements as there are input elements.

Full connectivity



When using a Kohonen network (SOM) there are always two phases

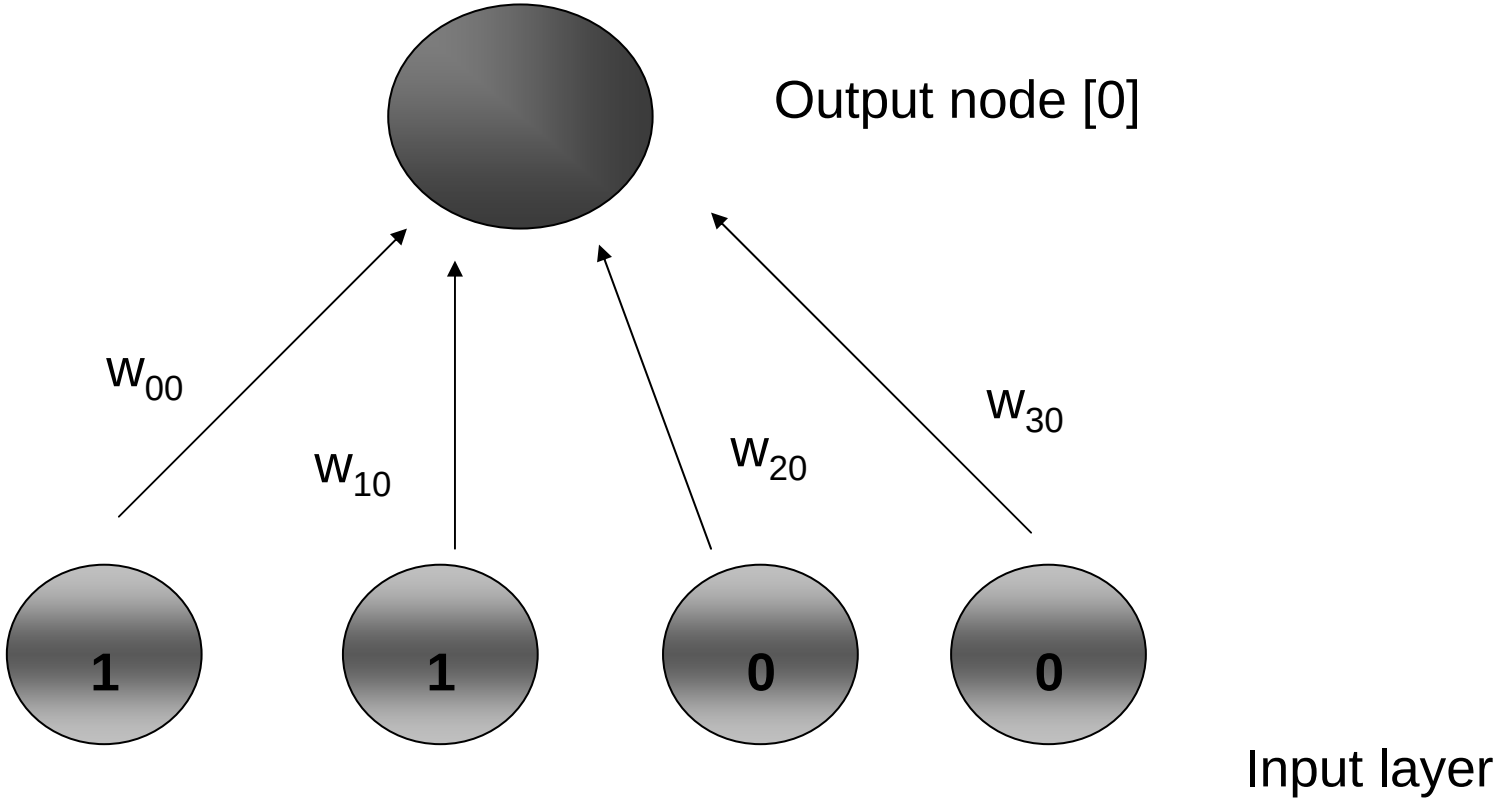
- Training phase – synaptic weights are changed to match input
- Testing phase – synaptic weights stay fixed. We test the performance of the network.

TRAINING PHASE

Training phase

Each node becomes associated with one input pattern or one category of inputs through a process of weight change. The weights are gradually changed to more resemble the input vector that it has finally come to represent.

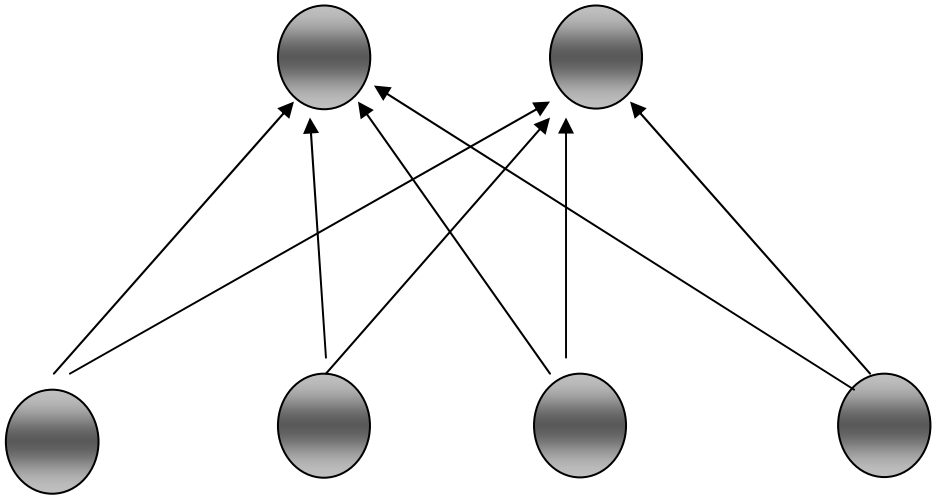
Step 1 – a matching process



$d_k = \Sigma (x_i - w_{ik})^2$ Euclidian distance

x_i is the input vector

Step 2 – choose a winner



Winner = d_{min} = min(d0,d1)

Weight update

Weights are updated for winner node and (the closest neighbours). The weights for the winner node are changed to more closely resemble the input vector.

Weight change Δw_{ij}

Δw_{ij} is proportional to $(x_{ij} – w_{ij})$

Δw_{ij} is proportional to R (spatial parameter that ensures biggest changes are further away)

Δw_{ij} is proportional to α (temporal parameter that ensures that biggest weight changes occur at the start of the training)

Full equation for weight update

$$\Delta w_{ij}(t) = \alpha(t)\, f(R)\, (x_i(t) - w_{ij}(t))$$

learning rate

learning radius

$$\alpha(t+1) = 0.5\alpha(t)$$

Some networks use more complex functions for the weight update

$$\Delta w_{ij}(t) = \eta(t) \quad [\exp(-d^2(i,winner)/2\sigma^2(t)] \quad (x_i(t) - w_{ij}(t))$$

Biggest weight changes occur at the beginning of training

$\eta(t) = \eta_i(\eta_f/\eta_i)^{t/T} \quad 0<t<T$

Biggest weight changes occur for the winner and for the closest nodes.

$\sigma(t) = \sigma_i(\sigma_f/\sigma_i)^{t/T} \quad 0<t<T$

A Kohonen self-organizing map (SOM) to cluster four vectors

- The vectors to be clustered (1,1,0,0) (0,0,0,1) (1,0,0,0)(0,0,1,1)
- Clusters to be formed 2
- Initial learning rate $\alpha(0) = 0.6$
- Radius of learning R = 0
- For the two output vectors the initial weight matrix is randomly set between 0 and 1.

$$W_{ij} \quad \begin{bmatrix} .2 & .8 \\ .6 & .4 \\ .5 & .7 \\ .9 & .3 \end{bmatrix}$$

Begin training

For input vector (1,1,0,0)

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

$D(2) = (0.8-1)^2 + (0.4-1)^2 + (0.7-0)^2 + (0.3-0)^2 = 0.98$

The input vector is closest to output node 2, so the weights on the winning unit are updated

$w_{i2}(t+1) = w_{i2}(t) + 0.6(x_i - w_{i2}(t))$

This gives the synaptic weight matrix

$$\begin{bmatrix} .2 & .92 \\ .6 & .76 \\ .5 & .28 \\ .9 & .12 \end{bmatrix}$$

Training with second vector

For input vector (0, 0, 0, 1)

$D(1) = (0.2-0)^2 + (0.6-0)^2 + (0.5-0)^2 + (0.9-1)^2 = 0.66$

$D(2) = (0.92-0)^2 + (0.76-0)^2 + (0.28-0)^2 + (0.12-1)^2 = 2.2768$

The input vector is closest to output node 1, so the weights on the winning unit are updated

$w_{i1}(t+1) = w_{i1}(t) + 0.6(x_i - w_{i1}(t))$

This gives the synaptic weight matrix

$$\begin{bmatrix} .08 & .92 \\ .24 & .76 \\ .20 & .28 \\ .96 & .12 \end{bmatrix}$$

Training with third vector

For input vector (1, 0, 0, 0)

$D(1) = (0.08-1)^2 + (0.24-0)^2 + (0.2-0)^2 + (0.96-0)^2 = 1.86$

$D(2) = (0.92-1)^2 + (0.76-0)^2 + (0.28-0)^2 + (0.12-0)^2 = 0.67$

The input vector is closest to output node 1, so the weights on the winning unit are updated

$w_{i1}(t+1) = w_{i1}(t) + 0.6(x_i - w_{i1}(t))$

This gives the synaptic weight matrix

$$\begin{bmatrix} .08 & .968 \\ .24 & .304 \\ .20 & .112 \\ .96 & .048 \end{bmatrix}$$

Training with the fourth vector

For input vector (0,0,1,1)

$D(1) = (0.08-1)^2 + (0.24-0)^2 + (0.2-0)^2 + (0.96-0)^2 = 0.7056$

$D(2) = (0.968-1)^2 + (0.304-0)^2 + (0.112-0)^2 + (0.048-0)^2 = 2.724$

The input vector is closest to output node 1, so the weights on the winning unit are updated

$w_{i1}(t+1) = w_{i1}(t) + 0.6(x_i - w_{i1}(t))$

This gives the synaptic weight matrix

$$\begin{bmatrix} .032 & .968 \\ .096 & .304 \\ .680 & .112 \\ .984 & .048 \end{bmatrix}$$

We have completed one cycle of learning with all four input vectors

Now it is time to reduce the learning rate

$\alpha = 0.5(0.6) = 0.3$

The weight update equations are now

$w_{ij}(new) = w_{ij}(old) + 0.3[\, xi - w_{ij}(old) \,]$

Iteration 0

$$\begin{bmatrix} .2 & .8 \\ .6 & .4 \\ .5 & .7 \\ .9 & .3 \end{bmatrix}$$

Iteration 1

$$\begin{bmatrix} .032 & .970 \\ .096 & .300 \\ .680 & .110 \\ .980 & .048 \end{bmatrix}$$

Iteration 2

$$\begin{bmatrix} .0053 & .990 \\ -0.17 & .300 \\ .700 & .02 \\ 1.00 & 0.0086 \end{bmatrix}$$

Iteration 10

$$\begin{bmatrix} 1.5e-7 & 1.00 \\ 4.6e-7 & .370 \\ .630 & 5.4e-7 \\ 1.000 & 2.3e-7 \end{bmatrix}$$

Iteration 50

$$\begin{bmatrix} 1.9e-19 & 1.000 \\ 5.7e-15 & 0.47 \\ .53 & 6.6e-15 \\ 1.000 & 2.8e-15 \end{bmatrix}$$

Iteration 100

$$\begin{bmatrix} 6.7e-17 & 1.000 \\ 2.0e-16 & .49 \\ .5100 & 2.3e-16 \\ 1.000 & 1.0e-16 \end{bmatrix}$$

The weight matrix has converged to the following matrix

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 0.5 \\ 0.5 & 0.0 \\ 1.0 & 0.0 \end{bmatrix}$$

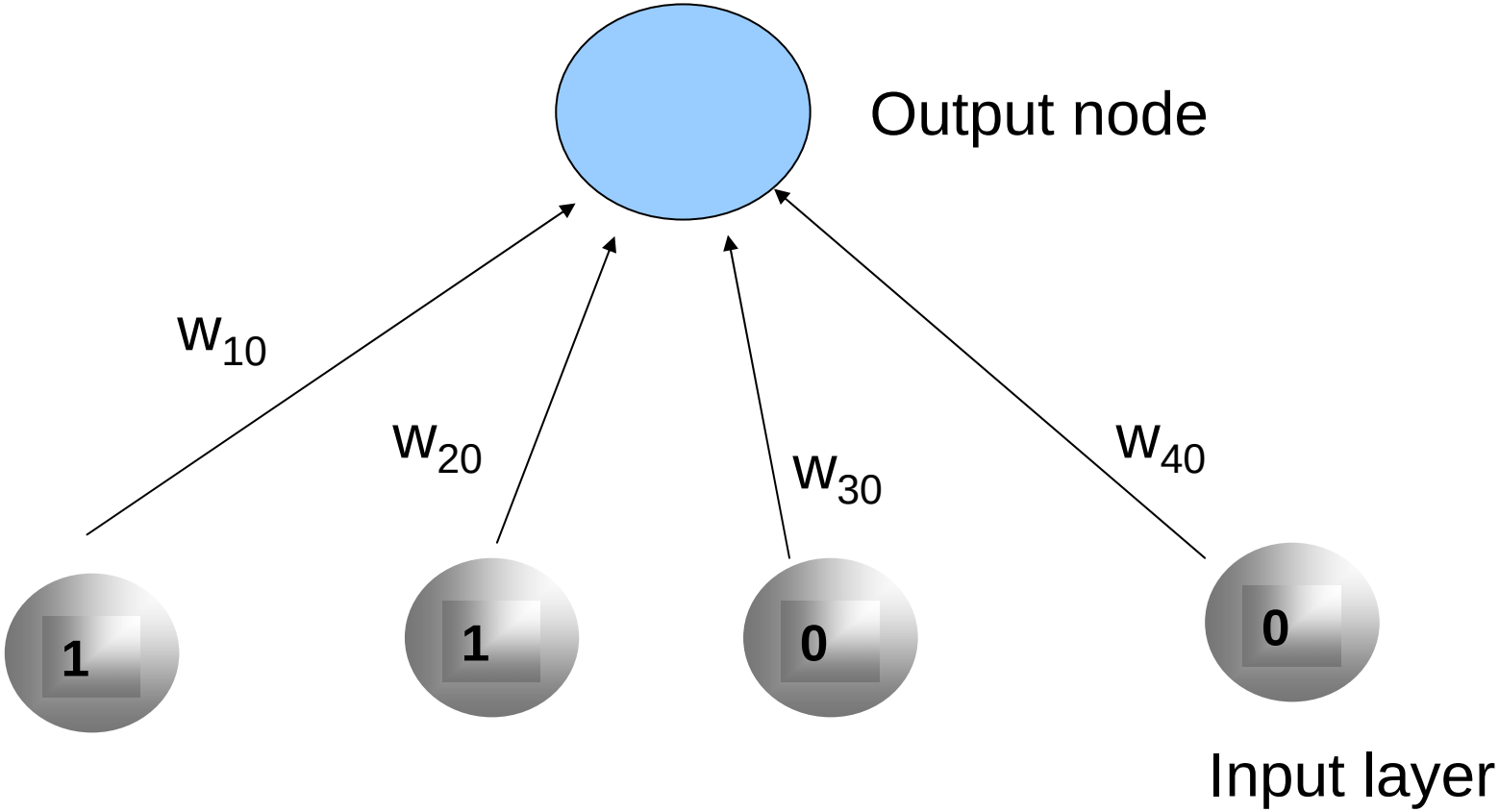
The first column is the average of the two vectors placed in cluster 1 and the second column is the average of the two vectors placed in cluster 2.

TESTING PHASE

In the testing phase the weights are fixed. With the weights that were obtained in the training phase, you can now test each vector. If the training was successful, the right node should light up for the right input vector.

In the testing phase, the Euclidian distance is calculated between input vector and weight vector. The node with the shortest distance is the winner.

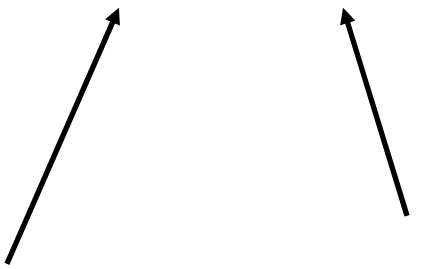
Testing phase - matching process



$d_k = \Sigma\ (x_i - w_{ik})^2$ Euclidian distance

The input vectors are classified based on their distance from the weight matrix of the training phase

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 0.5 \\ 0.5 & 0.0 \\ 1.0 & 0.0 \end{bmatrix}$$



(0, 0, 0, 0.9)

(0, 0, 0.8, 0.9)

(0.7, 0, 0, 0)

(0.7, 0.9, 0, 0)

Just a fake example. Things should become clearer with a real application

e.g. the classification of characters with different fonts.

Font 1

A B C D E J K

Font 2

A B C D E J K

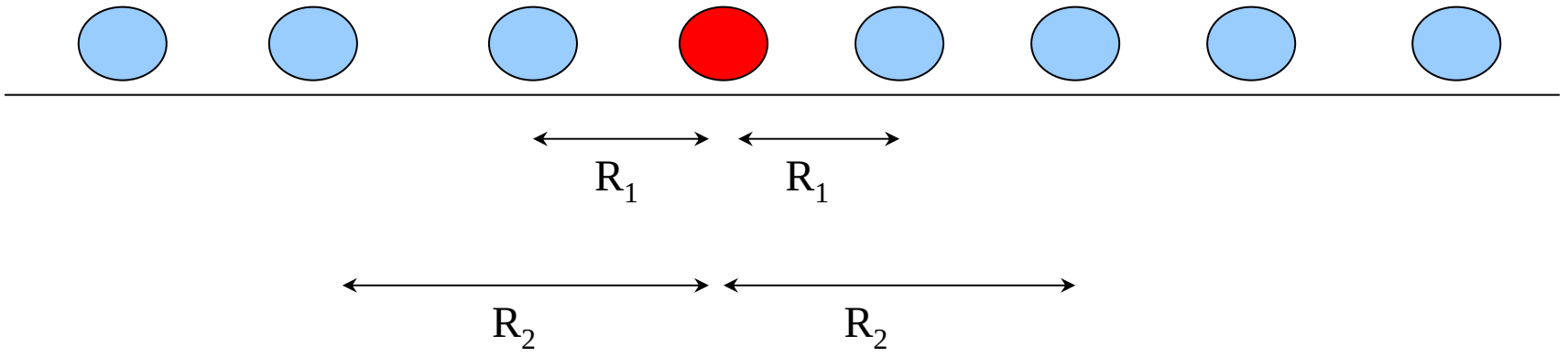
Font 3

A B X Δ E ϑ K

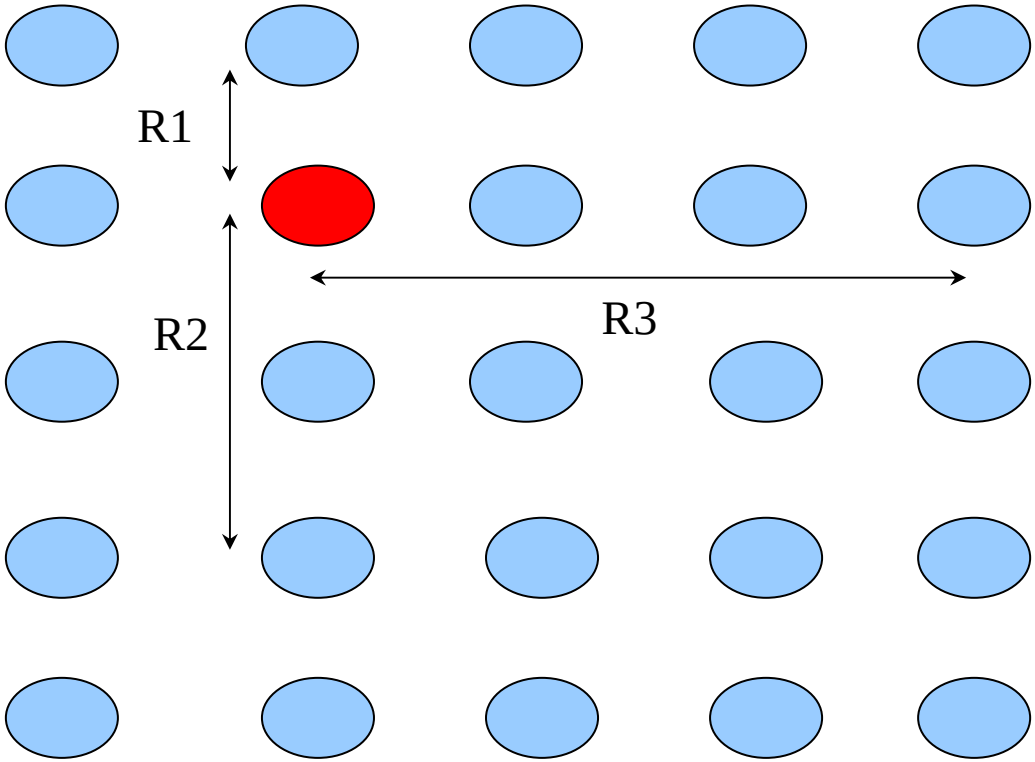
The role of network geometry

Linear organization

Output layer



2 D Organization



The role of network geometry

A Kohonen network with 25 nodes was used to classify the 21 characters with different fonts. In each example the learning rate was reduced linearly from 0.6 to a final value of 0.01. We will take a look at the role of network geometry in the classification.

A SOM to cluster letters from different fonts: no topological structure

If no structure is assumed for the cluster units i.e. if only the winning unit is allowed to learn the pattern presented, the 21 patterns form 5 clusters

UNIT	PATTERNS
3	C1, C2, C3
13	B1, B3, D1, D3, E1, K1, K3
16	A1, A2, A3
18	J1, J2, J3
24	B2, D2, E2, K2

A SOM to cluster letters from different fonts: linear structure

A linear structure (with R=1) gives a better distribution of patterns

UNIT	PATTERNS	UNIT	PATTERNS
6	K2	20	C1, C2, C3
10	J1, J2, J3	22	D2
14	E1, E3	23	B2, E2
16	K1, K3	25	A1, A2, A3
18	B1, B3, D1, D3		

A SOM to cluster letters from different fonts: 2D structure

i/j	1	2	3	4	5
1		J1,J2,J3		D2	
2	C1,C2,C3		D1,D3		B2,E2
3		B1		K2	
4			E1,E3,B3		A3
5		K1,K3		A1,A2	

The use of neural networks in order to analyze kinematic data from 3D gait analysis

Amount of data for each subjects

10 markers for two legs.

200 data points per marker per locomoter cycle i.e. 2000 data points for the legs per locomotor cycle.

Each person is recorded for approximately 10 cycles for 4 different types of gait (straight ahead and backwards on a straight or backward path).

Total 80.000 data points per person tested.

Many neuromuscular disorders create problems with gait

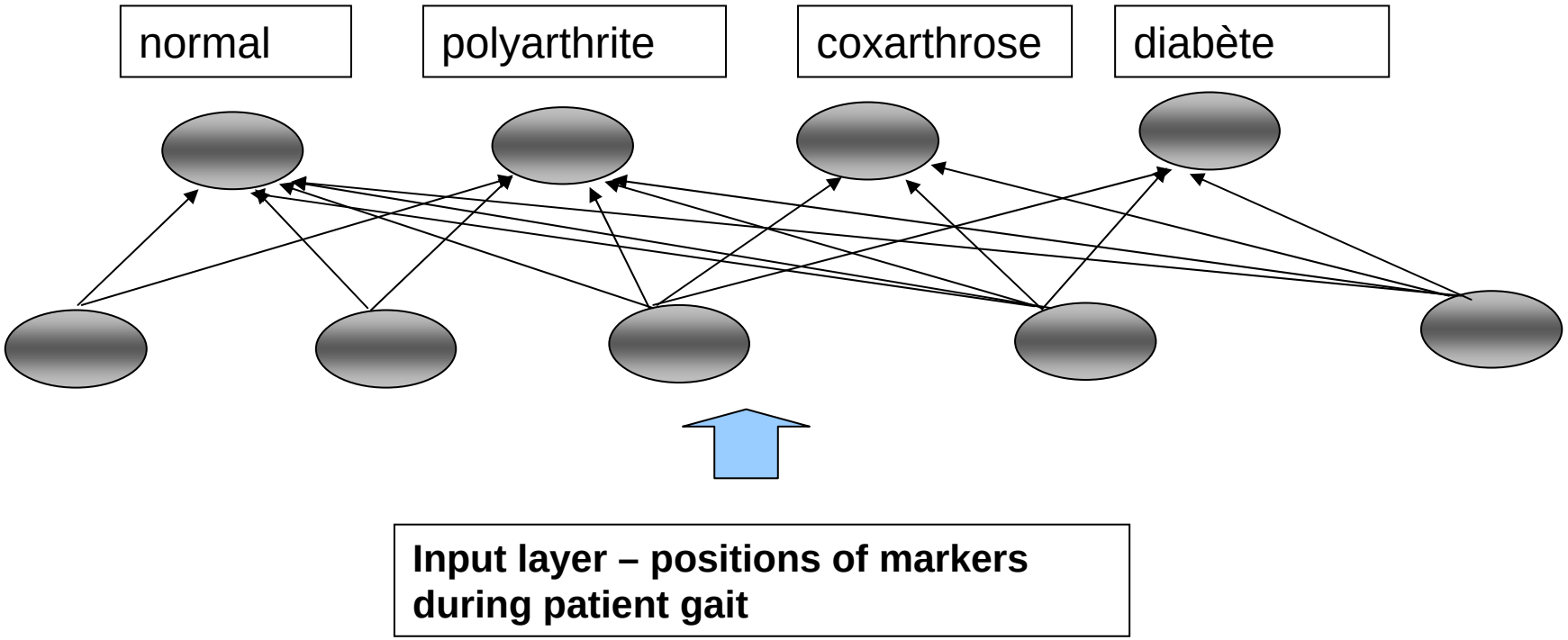
- Parkinsonism
- Various myopathies
- Peripheral nerve disorders (eg diabetes)
- Problems at the neuromuscular junction e.g. myasthenia gravis
- Cerebellar disorders

Missing data:

1. Markers that fall or that are displaced.
2. Movements that hide some markers.

The classification of locomotor disorders

An ongoing project at the laboratory INSERMU887 is the construction of neural networks in order to classify and to study locomotive disorders.



At the output layer we should be able to classify the disorders