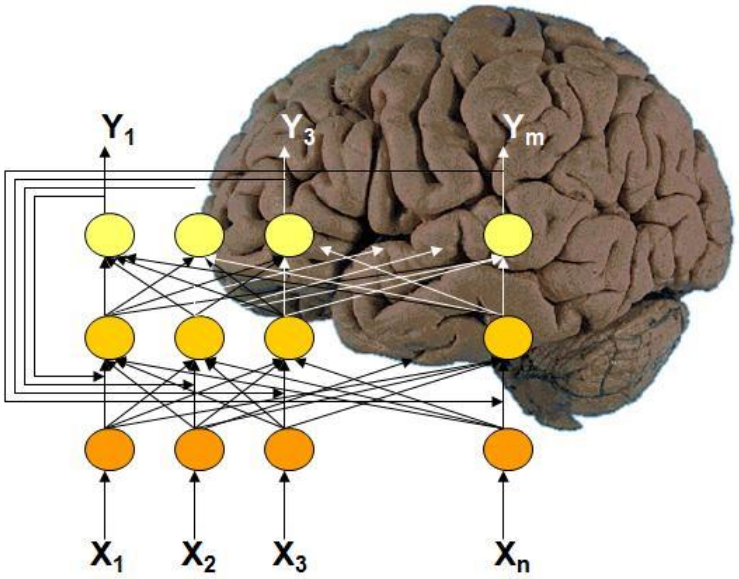
Artificial Neural Networks: Components, Models and Applications



Pedro Pablo González Pérez

Artificial Neural Networks: Components, Models and Applications

What, why and how / Historical background

Biological model

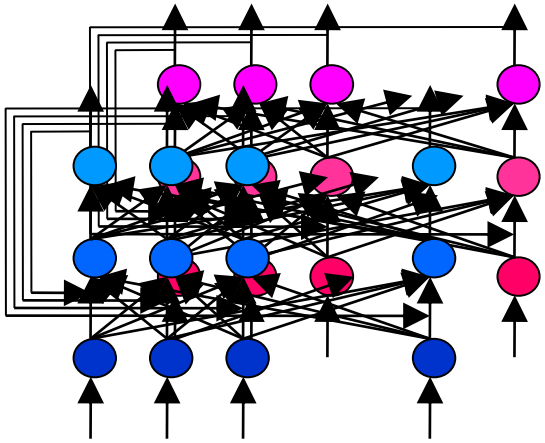
Artificial neuron

Neural network components

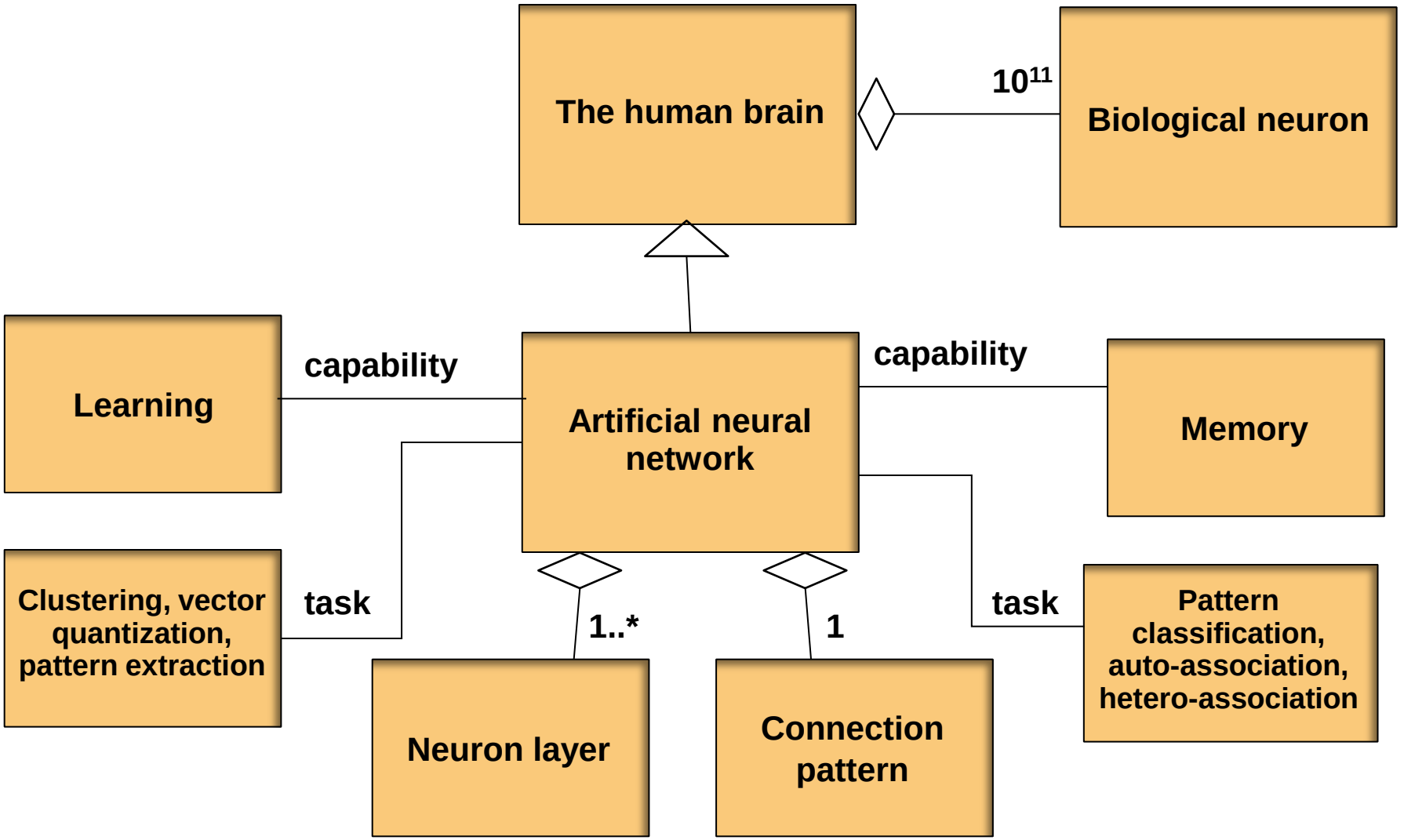
Neural network models

Neural network applications

Neural network tools



Artificial neural networks: What, Why and How



Neural networks: historical background

Years	Authors	Proposal
1943	McCulloch and Pitts	The first artificial model for biological neurons using binary threshold functions
1943	Landahl, McCulloch and Pits	Implementation of many arithmetic and logical operators using McCulloch and Pitts neuron model
1949	Hebb	The Hebbian learning rule
1956	Taylor	Associative memory network using Hebbian learning rule
1958	Rosenblatt	Perceptron: a learning method for the McCulloch and Pitts neuron model

Neural networks: historical background

Years	Authors	Proposal
1960	Widrow and Hoff	Adaline: a simple perceptron-like system trained by a gradient descendent rule to minimize mean squared error
1961	Rosenblatt	The backpropagation schema for training multilayer networks (unsuccessful)
1969	Minsky and Papert	Demonstration of the limits of simple perceptrons (linearly separable classes)
1986	Rumelhart et al.	The multilayer perceptron and backpropagation learning algorithm
Present	-	Many new neural network architectures, models and learning algorithms … and many, many applications

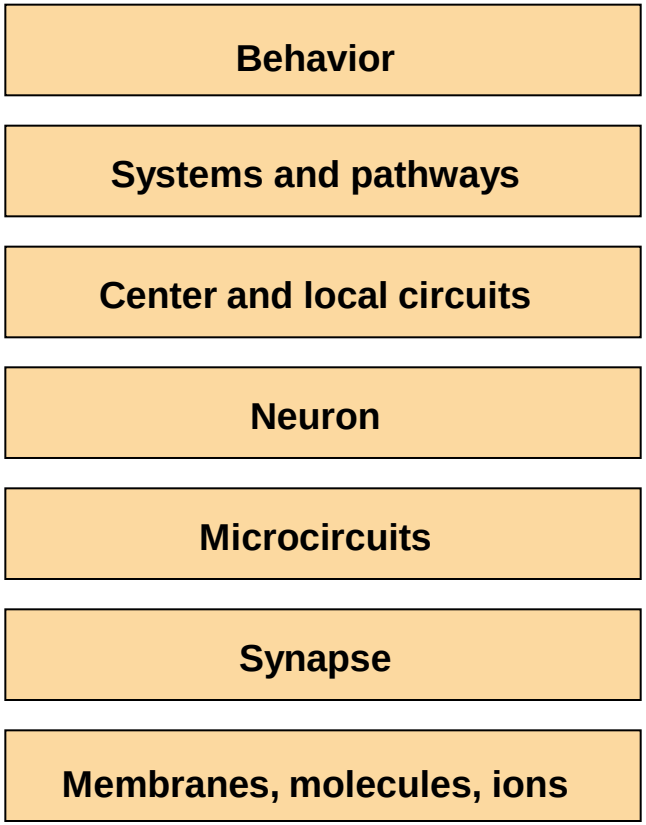
Biological model: the human brain

From an information–processing perspective, three levels can be distinguished in the human brain:

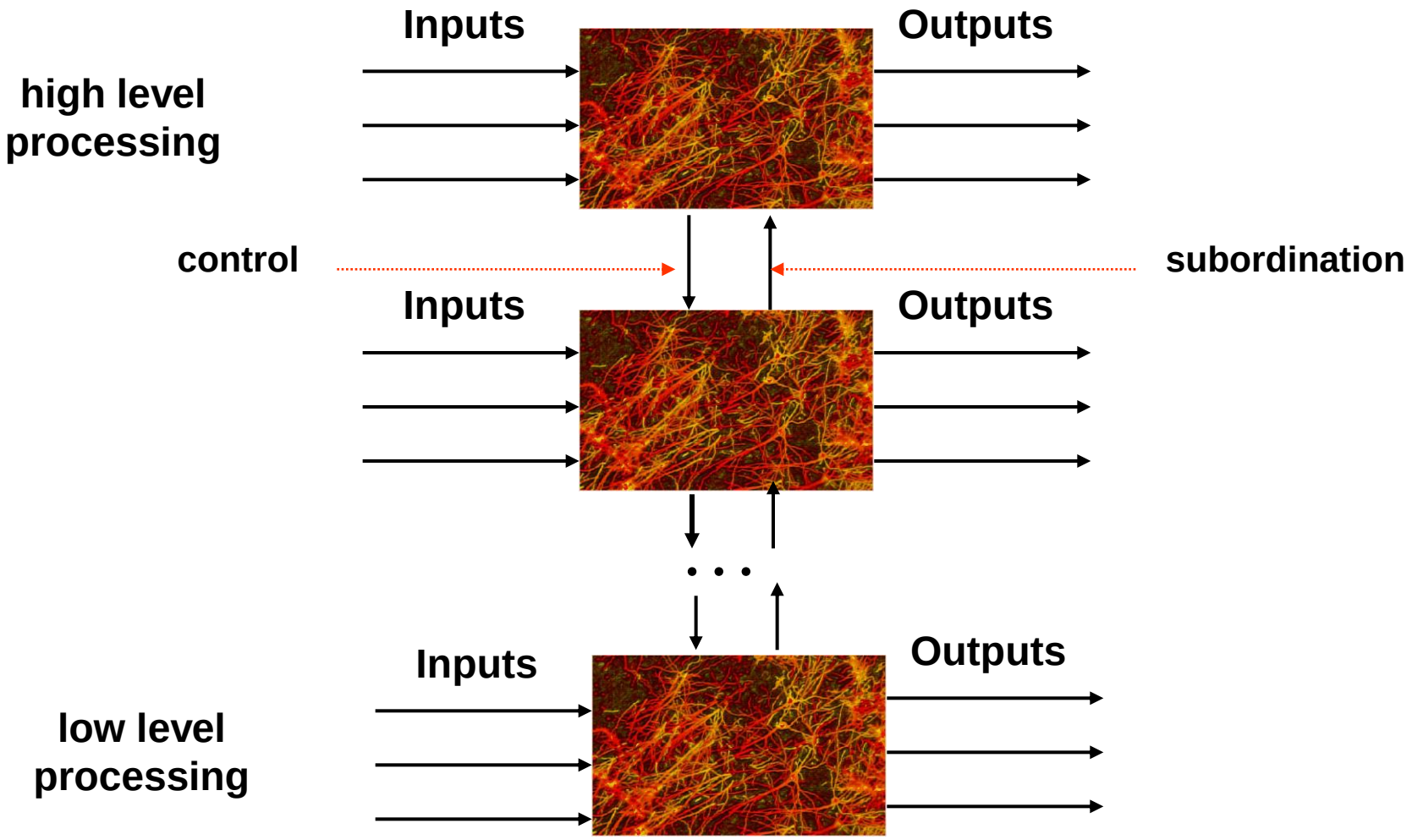
- **The structural level:** synapses, neurons, local networks, layers and columns, topographic maps, systems.
- **The physiological level:** chemical and physical reactions and transmission of substances.
- **The cognitive level:** human behavior

Biological model: the human brain

Levels of organization of the nervous system, as characterized by (Shepherd, 1988)



Biological model: the human brain

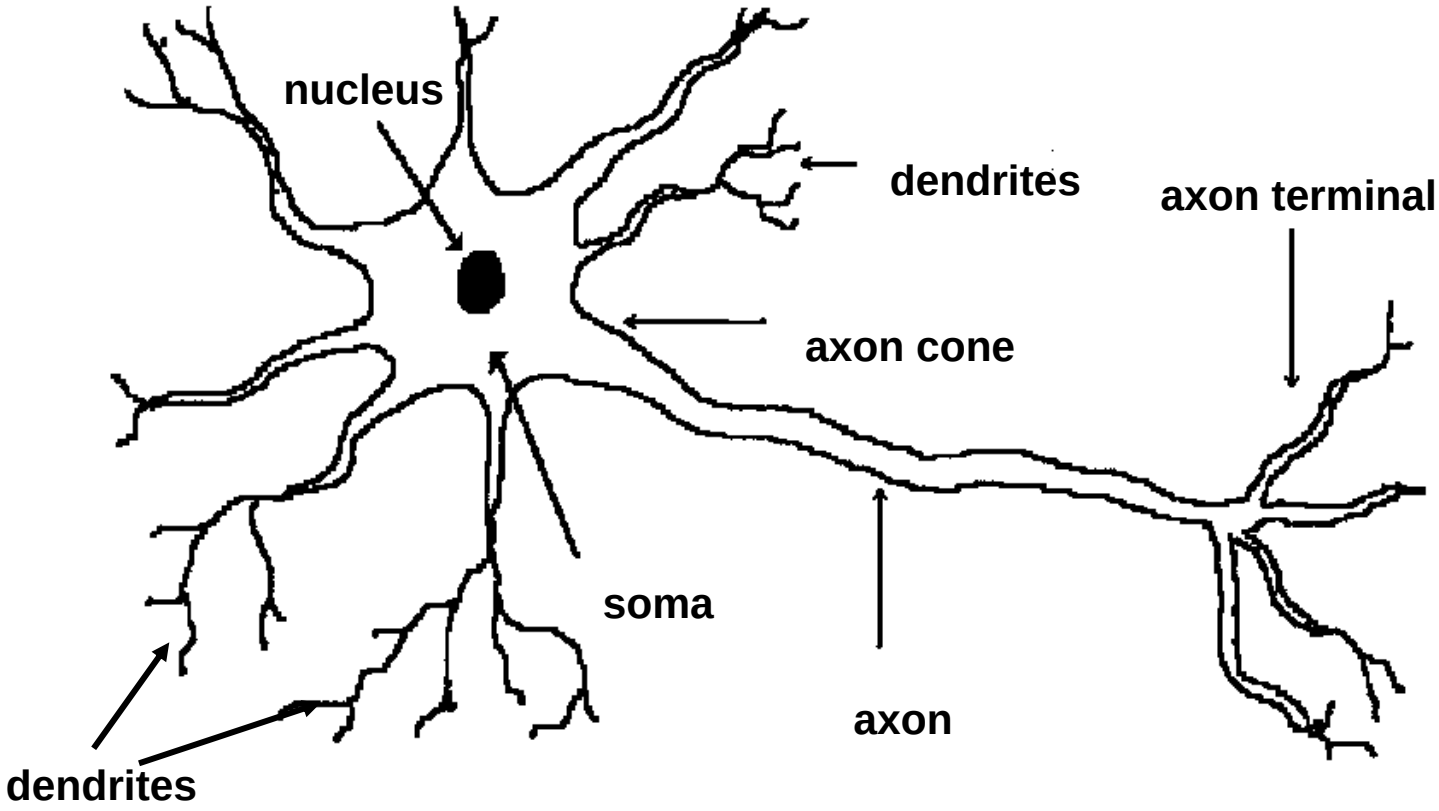


Biological model: the human brain

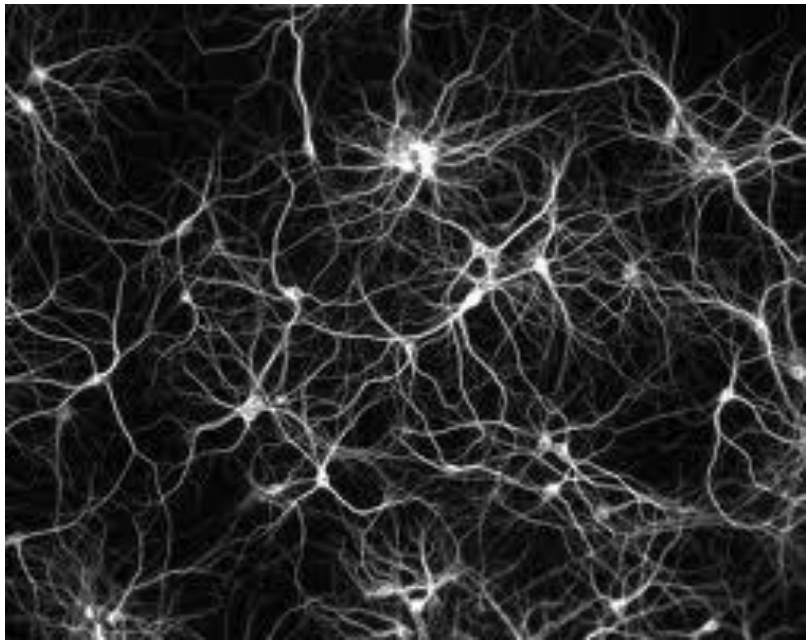
Structural features relevant to neural computation (Churchland and Sejnowski, 1992)

- Specialization of function
- Neurons and synapses
- Connectivity
- Analog inputs and discrete outputs
- Timing
- Cell-to-cell effects
- Firing patterns
- Receptive fields
- Specific and non specific systems
- Action-at-a-distance
- Parallel architecture

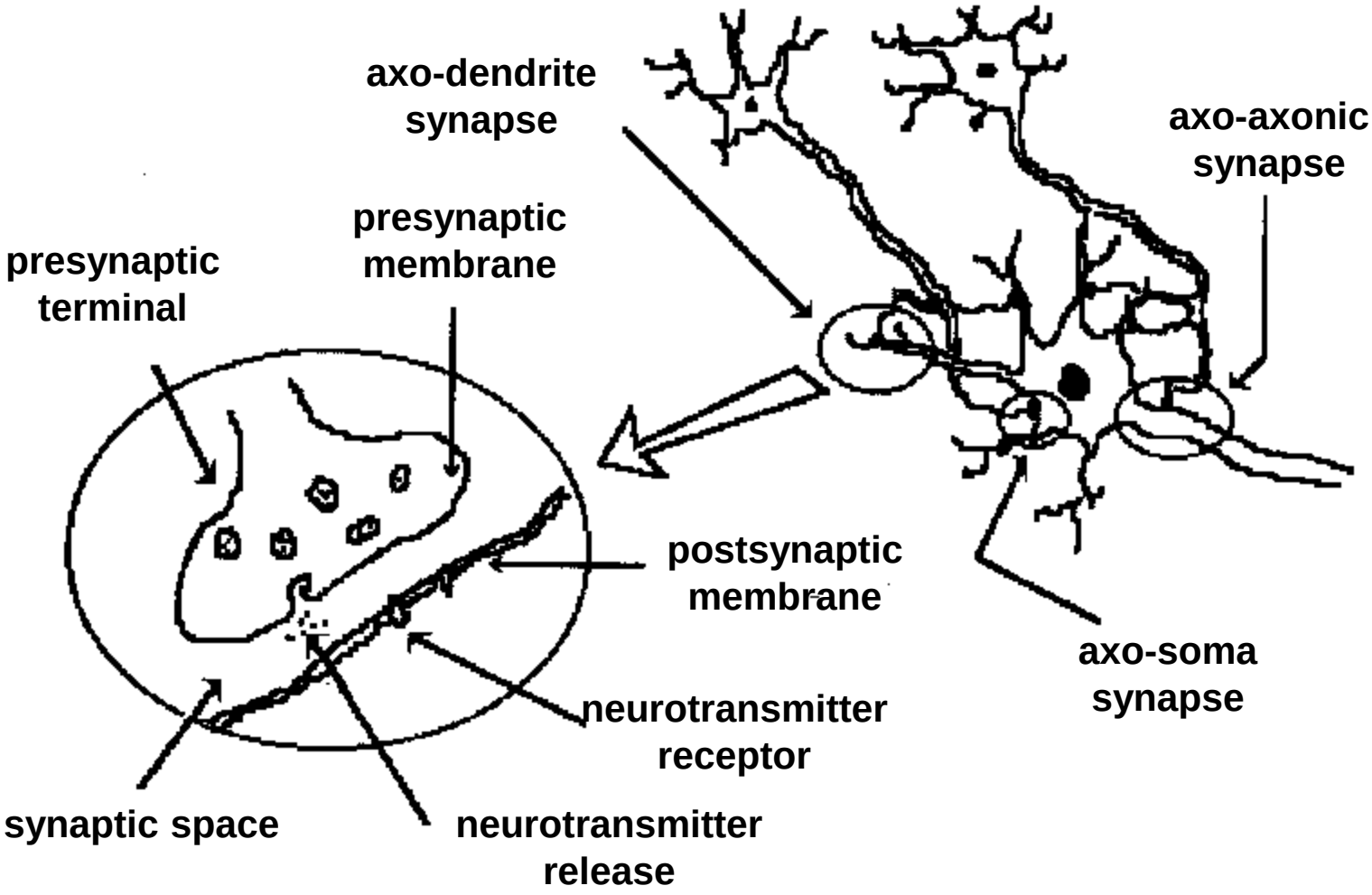
Biological neurons



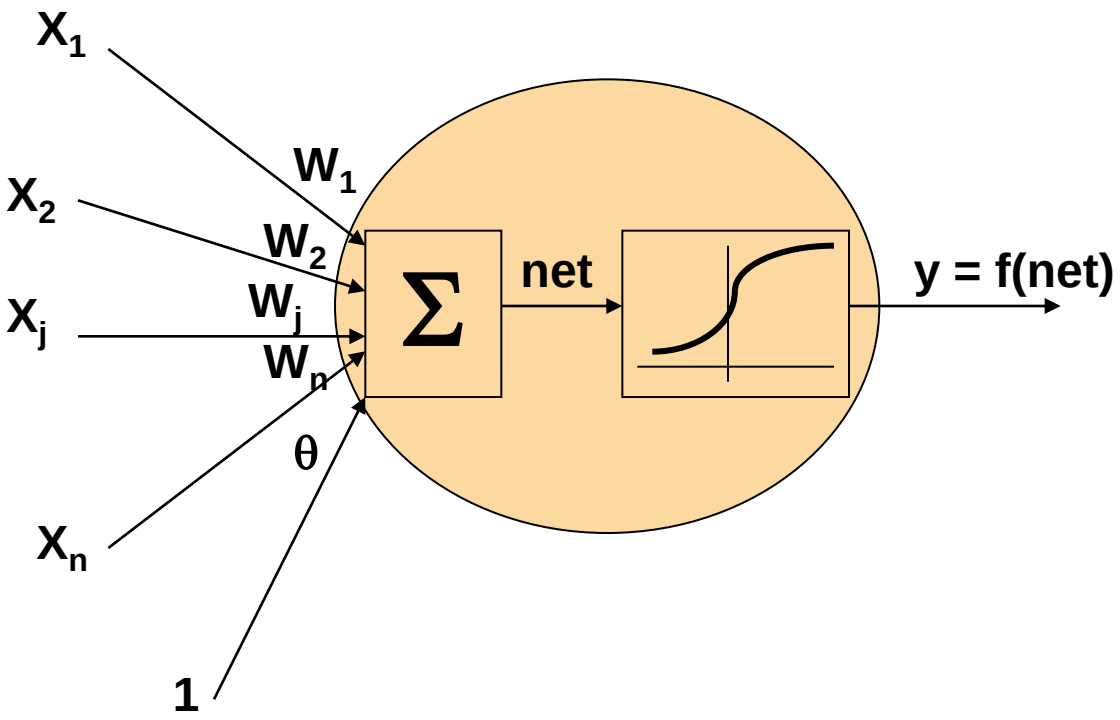
Biological neurons



Neurons and synapses



Artificial neuron



Input values: $X_1, X_2, \ldots, X_j, \ldots, X_n$

Synaptic weights: $X_1, X_2, \ldots, X_j, \ldots, X_n$

$net = \Sigma_j W_j X_j + \theta$

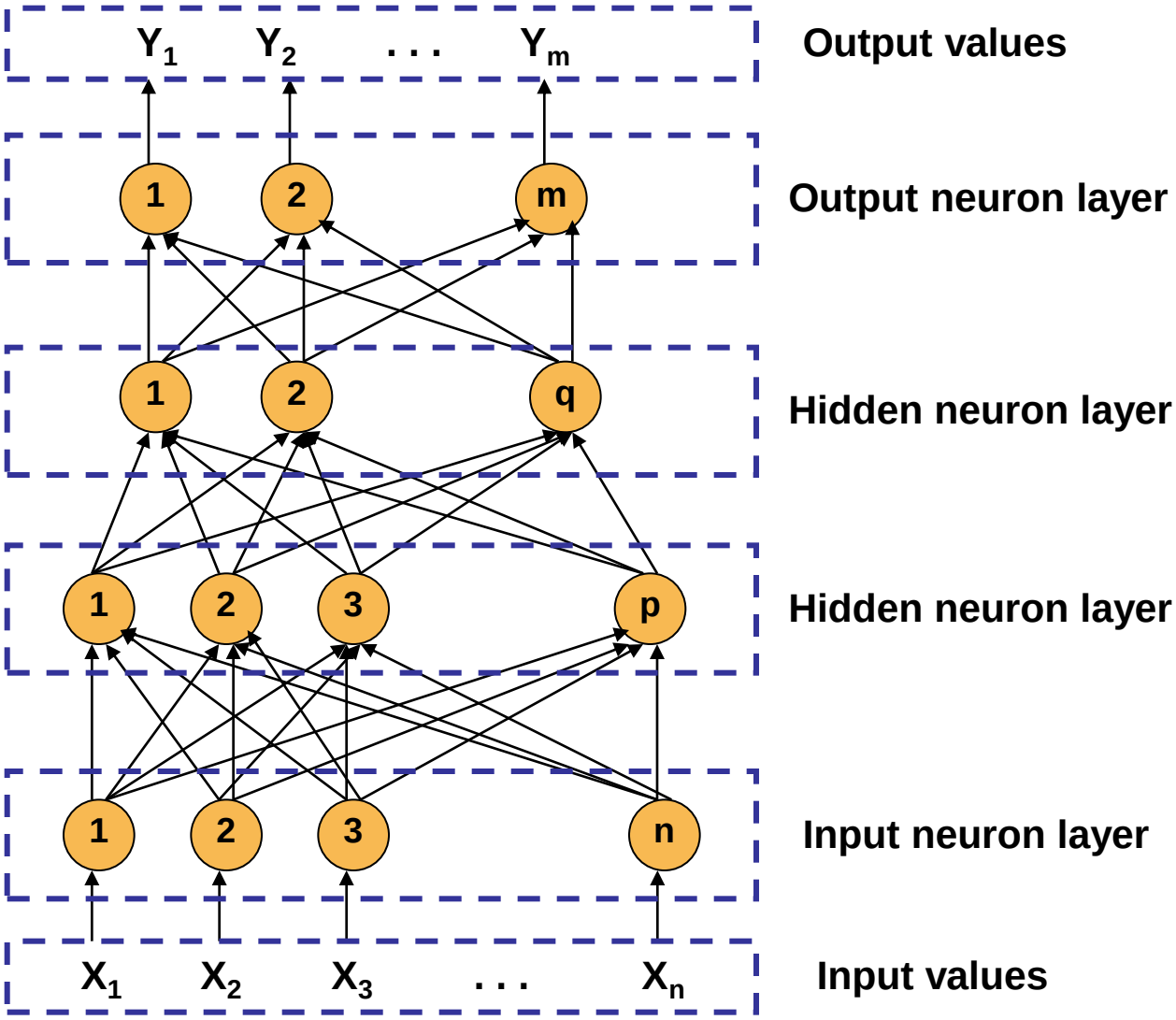
f(): activation function

y: output value

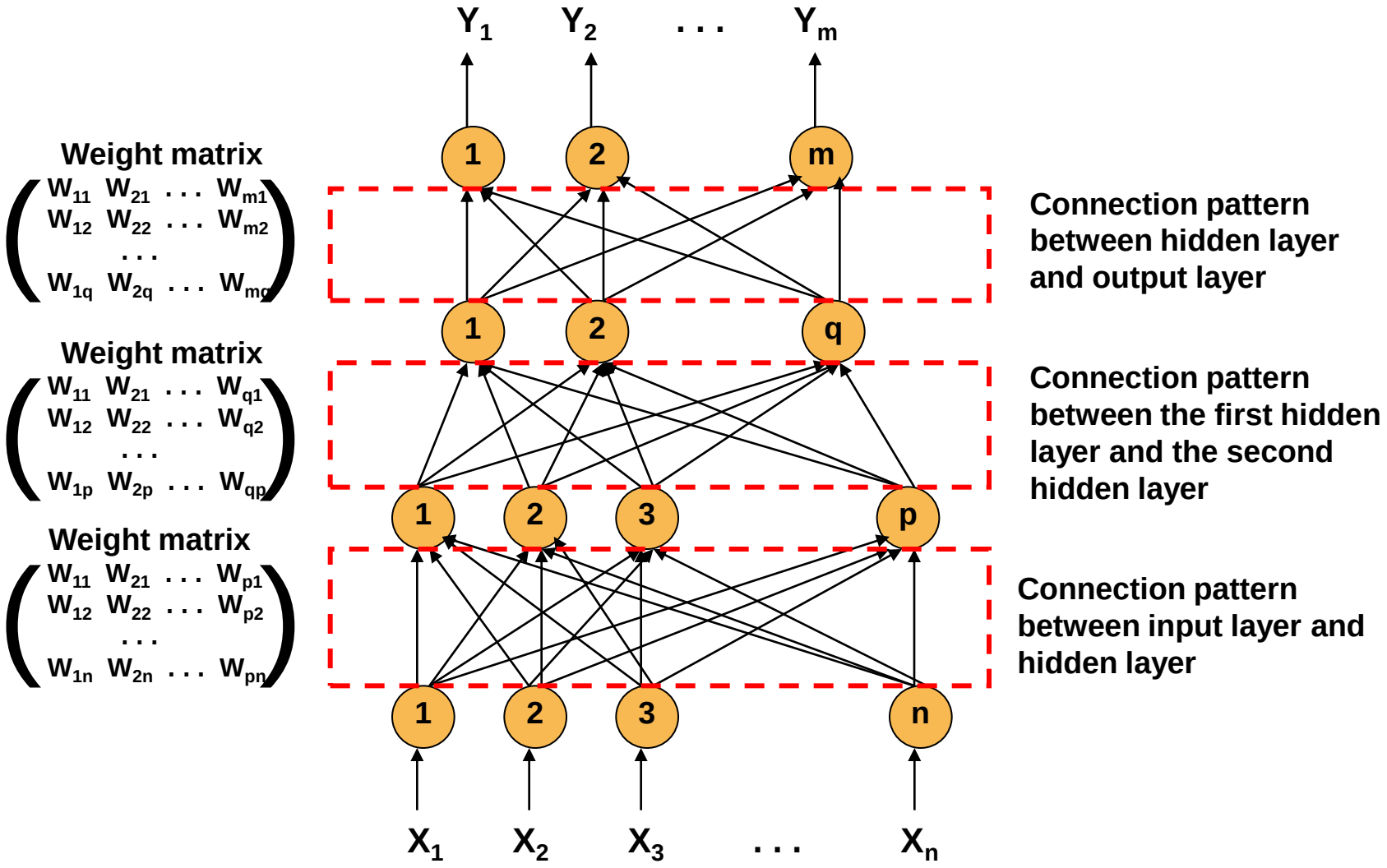
The components and functions of a neural network

- Neural network architecture:
  neuron layers and connection pattern
- Net functions
- Neuron activation functions
- The learning process

Neural network architecture: neuron layers and connection pattern

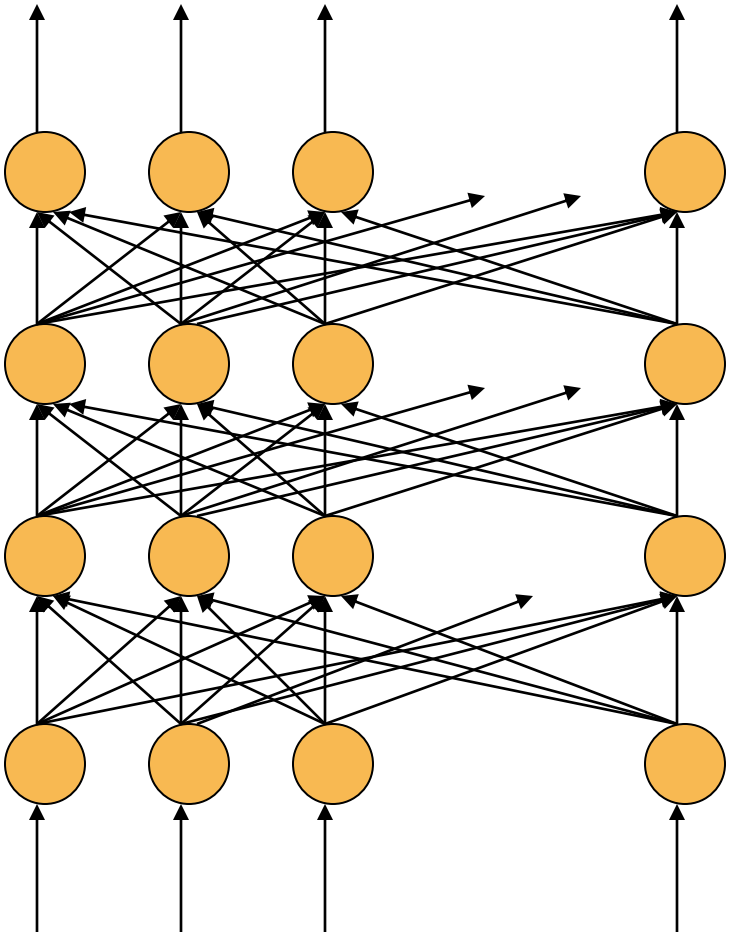


Neural network architecture: neuron layers and connection pattern



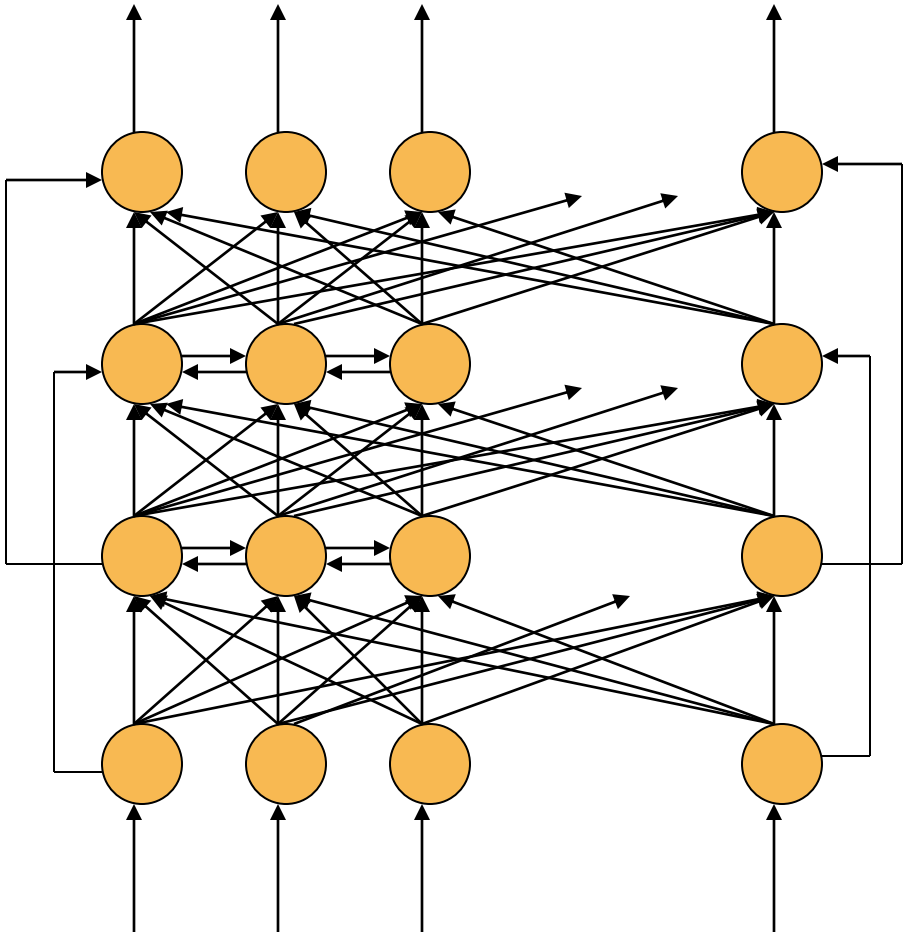
Neural network architecture: connection pattern (I)

Feedforward network: a connection is allowed from a neuron in layer i only to neurons in layer i+1.



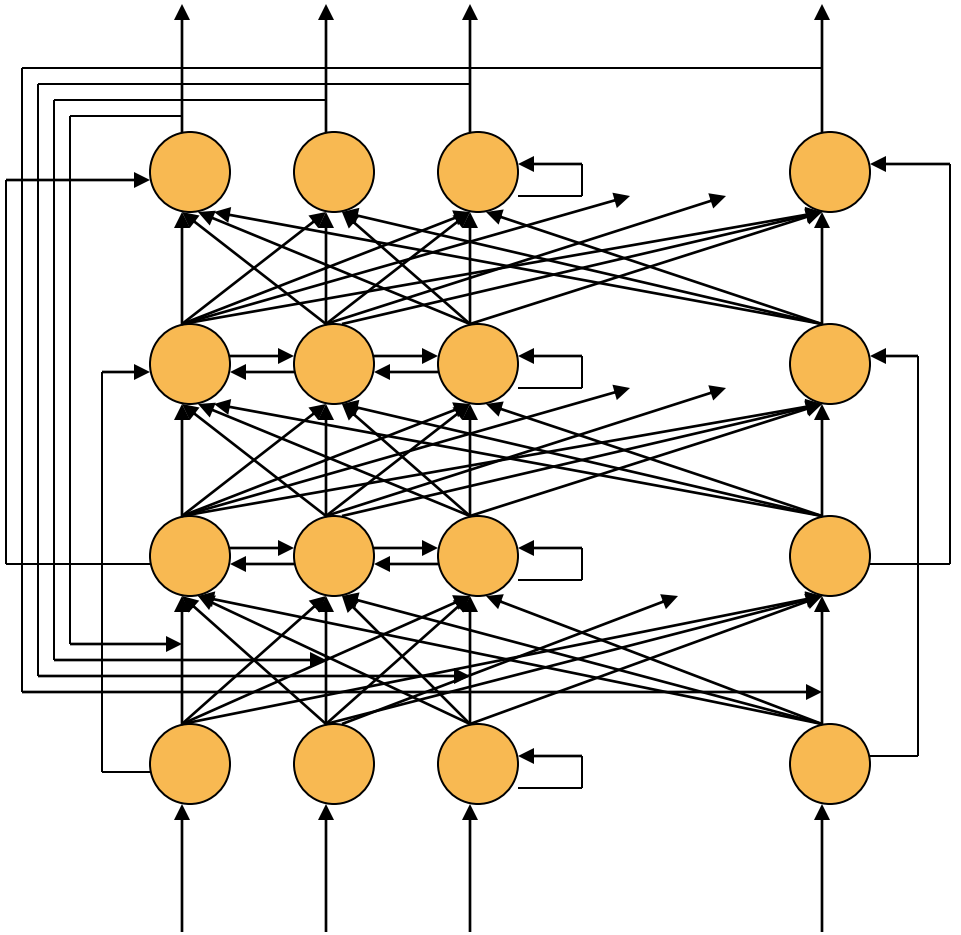
Neural network architecture: connection pattern (II)

In this type of network are no connections from a neuron in layer j to layer k if j > k. All others kinds of connections are allowed.



Neural network architecture: connection pattern (III)

A fully connected layered neural network: in this type of network every neuron can be connected to every neuron, and these connections patterns may be either feedforward, feedbackward, intralayer or feedback connection from the neuron to itself.



Net functions

The net function determines how the network inputs $\{X_j;\ 1 \leq j \leq N\}$ are combined inside the neuron. In almost all artificial neuron models all inputs X_j are weighted by the synaptic transmission efficiency $\{W_j;\ 1 \leq j \leq N\}$ and are summed. In this case, the neuron input consists of the value

$$net = \sum_{j=1}^{N} W_j X_j + \theta$$

The quantity θ is called the *bias* and is used to model the threshold.

Other net functions (less frequently used) are *Higher order* and *Delta* (Σ - Π)

Activation functions

Activation function	Formula
Identity function	$f(net) = net$
Hard-limited threshold function	$f(net) = \begin{cases} 1 \; if \; net > \theta \\ 0 \; if \; net < \theta \end{cases}$
Linear threshold function	$f(net) = \begin{cases} a & if \quad net < c \\ b & if \quad net > d \\ a + ((net - c)(b - a))/(d - c) & if \; otherwise \end{cases}$
Sigmoid function	$f(net) = \frac{1}{1 + e^{-net}}$
Hyperbolic tangent function	$f(net) = \tanh(net)$
Gaussian function	$f(net) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2} \left(\frac{net - u}{\sigma} \right)^2 \right)$

The learning process

- **Supervised learning**
  - **Perceptrons**
  - **Multilayer Perceptrons (MLP)**
- **Unsupervised learning**
  - **Noncompetitive learning**
  - **Competitive learning (winner takes all)**

The learning process

- **Perceptron learning rule (supervised learning)**

$$W_{jk}(t+1) = W_{jk}(t) + \eta\left(Y_j^*(t) - Y_j(t)\right)X_k(t)$$

- **Hebbian learning rule (unsupervised learning)**

$$W_{jk}(t+1) = W_{jk}(t) + \varepsilon Y_j(t)X_k(t)$$

Notation:

$j, k, \ldots$ the unit $j, k, \ldots$

X_k the output of the unit k

Y_j the output of the unit j

Y_j^* the desidered output of the unit j

W_{jk} the weight of the connection from unit k to unit j

$0 < \eta, \varepsilon < 1$ learning rates

Supervised learning

Major applications for supervised learning are:

- Classification
- Auto-associative memory
- Hetero-associative memory

Unsupervised learning

When unsupervised is used, the network attempts to discover special features and patterns from input data without using external teach signal. Major applications for unsupervised learning are:

- Clustering
- Vector quantization
- Probability distribution
- Feature extraction

Models of artificial neural networks

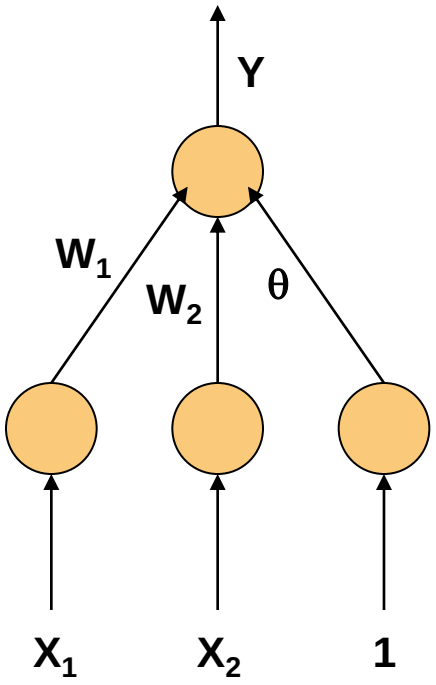
- Perceptron
- Multi-Layer Perceptron
- Recurrent Backpropagation Network
- Winner-Take-All Networks
- Hopfield Networks
- Kohonen Self Organizing Maps

Overview of Perceptron

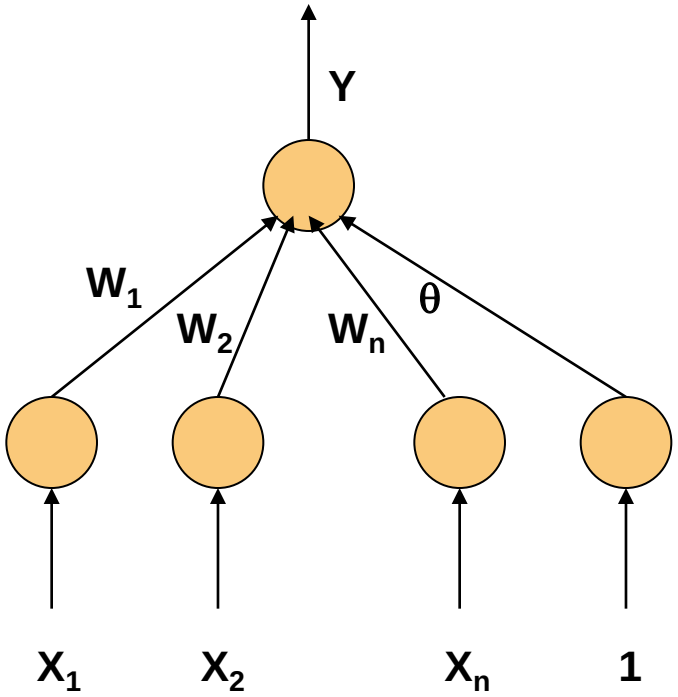
- A perceptron can be represented by an single neuron with a linear weighted net function and a threshold activation function.
- Perceptron = McCulloch and Pitts' neuron + supervised learning algorithm.
- The values of the single neuron determine to which of two classes each input pattern belongs.
- The correct value of the weight vector W is estimated using a sequential learning algorithm (perceptron learning algorithm).

Perceptron architecture

A simple two-inputs, one-output perceptron



Generic perceptron for n-dimensional input space



$$net = \sum_{j=1}^{n} W_j X_j + \theta \qquad f(net) = \begin{cases} 1 & if\ net > 0 \\ 0 & \text{otherwise} \end{cases}$$

Perceptron characteristics

Neuron layers	One input layer One output layer
Connection pattern	Feedforward
Input value types	Real
Activation function	Hard limiter
Learning method and learning algorithm	Supervised $W_{jk}(t+1) = W_{jk}(t) + \eta\left(Y_j^*(t) - Y_j(t)\right)X_k(t)$

Multilayer perceptron (MLP) and backpropagation algorithm: overview

- A multilayer perceptron is a feed-forward, layered network, containing one input layer, at least one “hidden” layer, and one output layer.
- Input layer neurons only transmit the input pattern to the hidden layer neurons, do not perform any computation.
- The neurons in the output and hidden layers have continue values inputs and outputs, a net function and a nonlinear activation function.
- The backpropagation algorithm is an implementation of the least mean squared technique, that modifies network weights to minimize the mean squared error between the desired and actual outputs.

Multilayer perceptron architecture

Notation:

L : layer index

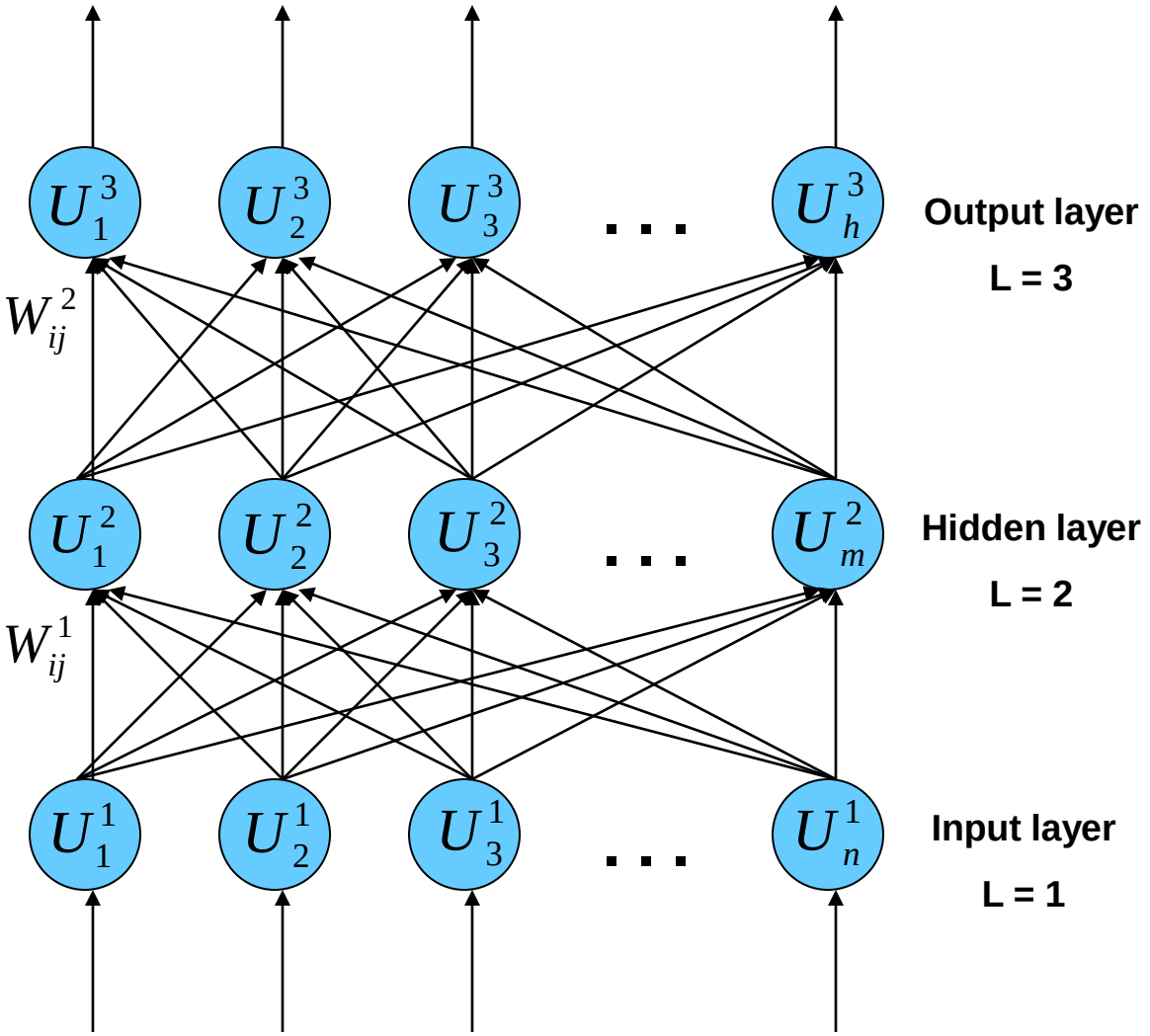
U_i^L : neuron i in the layer L

W_{ij}^L : the weight of the conexion from the neuron i in the layer L to the neuron j in the layer $L+1$

n : number of neurons in the input layer

m : number of neurons in the hidden layer

h : number of neurons in the output layer

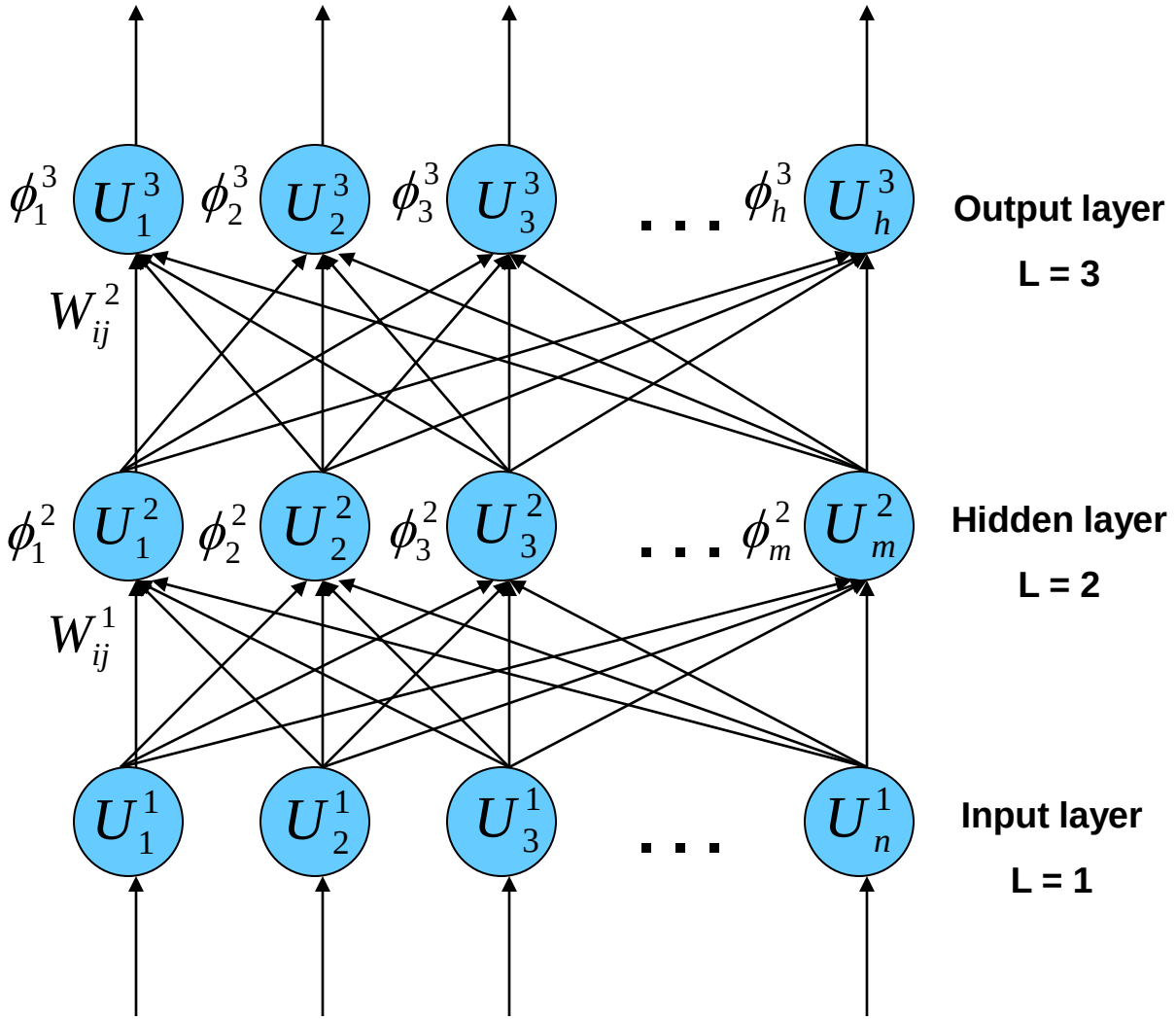


Multilayer perceptron characteristics

Neuron layers	One input layer One or more hidden layers One output layer
Connection pattern	Feedforward
Input value types	Real
Activation function	Sigmoid
Learning method and learning algorithm	Supervised Backpropagation algorithm

How the error backpropagation is computed

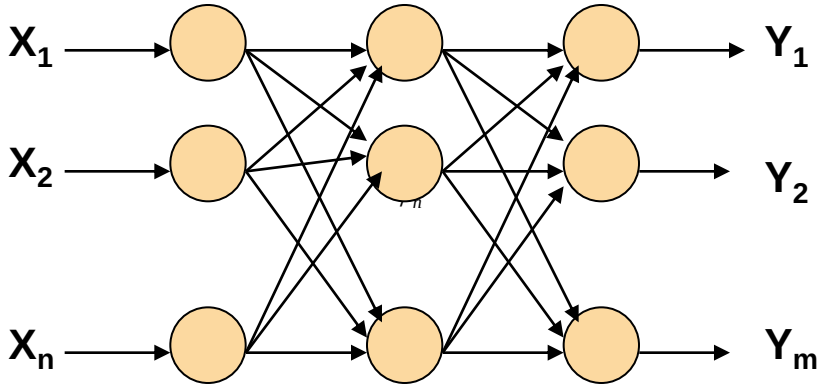
ϕ_i^L is iteratively computed from ϕ_j^{L+1} and weights in the layer $L+1$



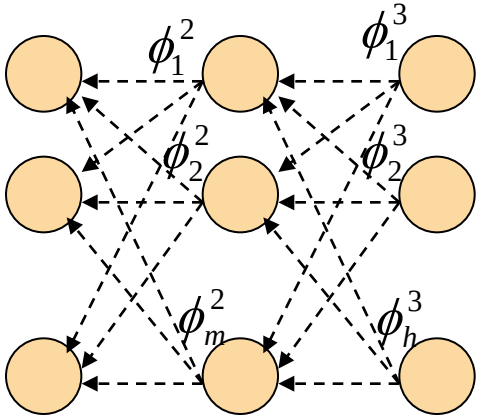
Forward propagation and backward propagation

- **Forward propagation**: the inputs are provided and propagated through the hidden layers to the output layer.
- **Backward propagation**: the error is calculated at the output layer and propagated backward through the hidden layers for calculating the changes of weights.

Forward propagation



Backward propagation



E (sum of square error)

The error backpropagation formula

$$E = \sum_{k=1}^{K} \left[e(k) \right]^2 = \sum_{k=1}^{K} \left[d(k) - y(k) \right]^2 \text{ is the sum of square error}$$

$$W_{ij}^{L,t+1} = W_{ij}^{L,t} + \alpha \Delta W_{ij}^{L}$$

$$\Delta W_{ij}^{L} \text{ is proportional to } -\frac{\delta E}{\delta W_{ij}^{L}}$$

$$-\frac{\delta E}{\delta W_{ij}^{L}} = \phi_j^{L+1} O_i^{L}$$

$$\text{where : } \phi_i^{L} = \begin{cases} \left(d_i^{L} - y_i^{L} \right) \left(F_i^{L} \left(A_i^{L} \right) \right)' & \text{if } L = \text{output layer index} \\ \left(F_i^{L} \left(A_i^{L} \right) \right)' \sum_{\text{j}} \phi_j^{L+1} W_{ij}^{L} & \text{otherwise} \end{cases}$$

$$W_{ij}^{L,t+1} = W_{ij}^{L,t} + \alpha \phi_j^{L+1} Y_i^{L}$$

F_i^{L} : activation function of the neuron i in the layer L

Backpropagation algorithm

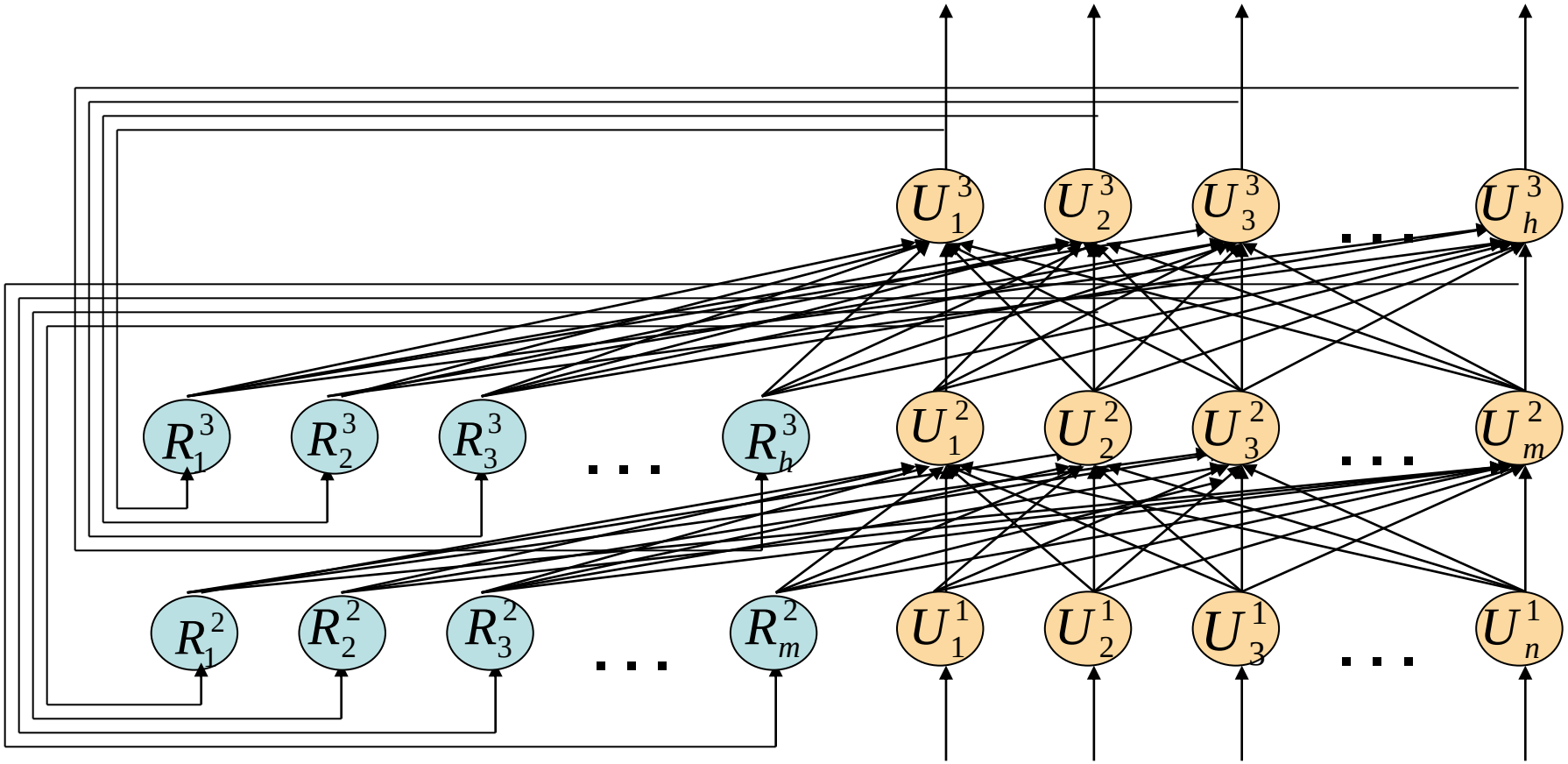
Setting parameter values:

- Weight initialization
- Weight updates
- Number of samples
- Learning rate
- Momentum
- Generalizability: error on test data vs. error on training data
- Number of hidden layers and neurons

Overview of Recurrent Neural Networks

- Recurrent neural networks have feedback connections. They contain connections from output layer neurons to hidden layer and/or input layer neurons.
- Recurrent neural networks address the temporal relationship of input patterns by maintaining internal memory states (recurrent neurons in figure below).
- The training algorithm typically used is backpropagation through time algorithm.
- Recurrent neural networks have found great use in function approximation problems, such as forecasting and control.

Recurrent Neural Network architecture



U Real neuron

R Recurrent neuron (internal memory state)

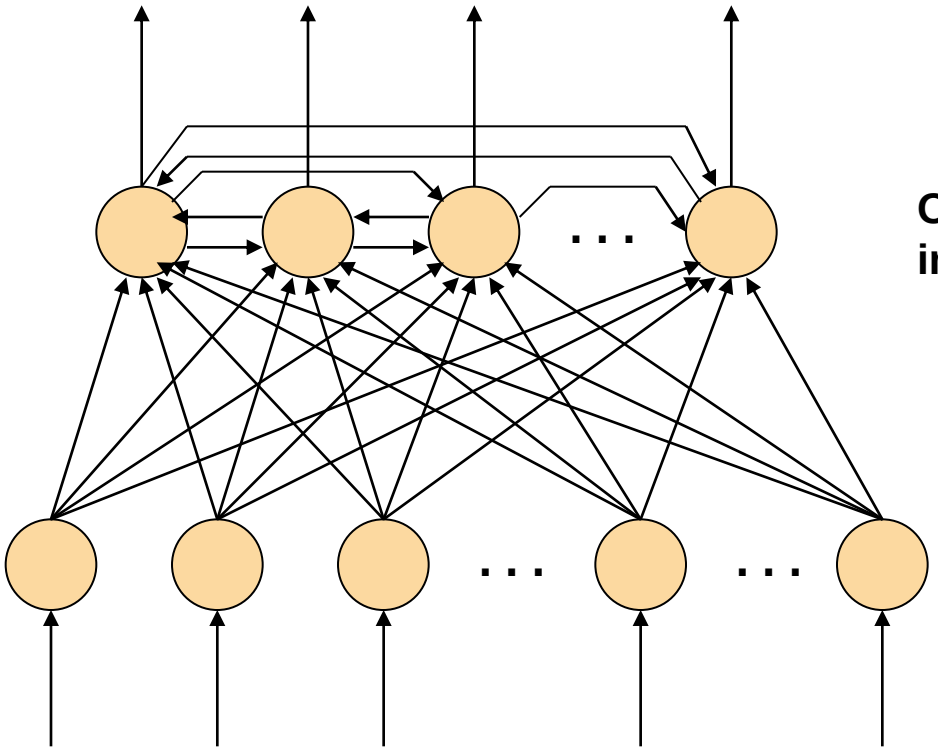
Recurrent Neural Network characteristics

Neuron layers	One input layer One or more hidden layer One output layer
Connection pattern	Feedforward and feedbackward
Input value types	Real
Activation function	Sigmoid
Learning method and Learning algorithm	Supervised Backpropagation through time

Overview of Winner-Take-All Networks

- The neural network is organized in layers. There are excitatory connections between neurons from different layers and inhibitory connections between neurons in the same layer.
- The neurons in the output layer compete among themselves to represent the current input pattern, each one inhibiting the others with its current activation level.
- After competition the winning neuron is activated, and its weight vector will be adjusted to be closer to the input pattern.
- The synaptic weight are updated according to a generalized Hebbian learning rule.

Winner-Take-All Network architecture



Output layer with inhibitory connections

Input layer

Winner-Take-All Network characteristics

Neuron layers	One input layer One output layer with inhibitory connections
Connection pattern	Feed forward Inhibitory connections between neurons in the output layer
Input value types	Real binary
Net function Activation function	Distance measure Hard-limited function (commonly used)
Learning method Learning rule	Non supervised Generalized Hebbian learning rule

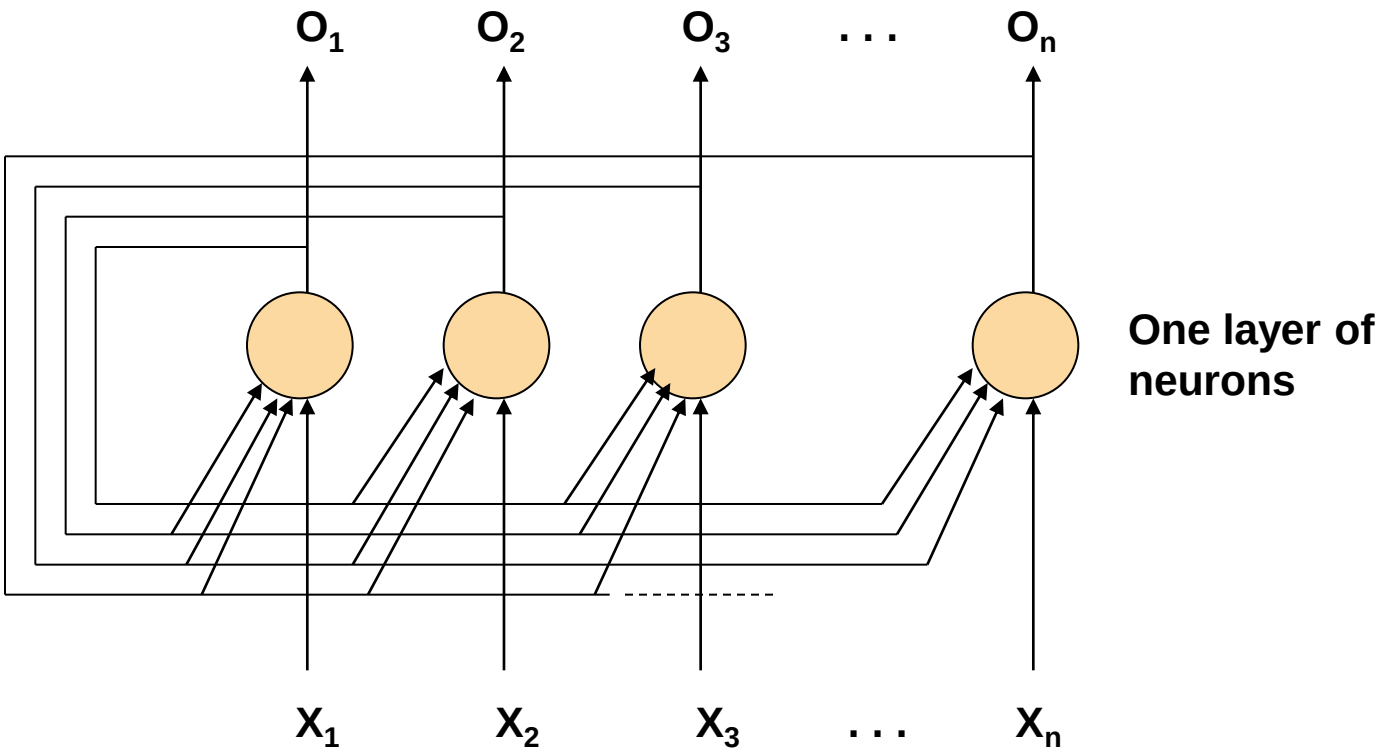
Overview of Hopfield Networks

- Hopfiled networks are neural networks commonly used for auto-association and optimization tasks.
- Hopflied networks are fully connected feedback networks: every neuron in the network is connected back to every other one, except itself.
- The weights are updated by the Hebbian rule.
- Each neuron in the network receives an input pattern composed of an external input and the weighted outputs of the remaining neurons.
- Each neuron in the network represents an attractor for a specific type of pattern.

Overview of Hopfield Networks

- When the network is used for auto-association problems, the desired output vector for a given input pattern consists of the nearest stored attractor pattern.
- Each neuron in the network performs the following two steps: (1) compute the sum of the its corresponding external input and the weighted outputs of the other neurons, (2) apply a non-linear function to the calculated sum, resulting in an output of either +1 or -1.
- When an input vector is presented to the network, the output values of the neurons may change until the state of the network converges to an specific attractor.

Hopfield Network architecture



The main process in the Hopfield Networks

- The storage of patterns in neural network
- The recall process

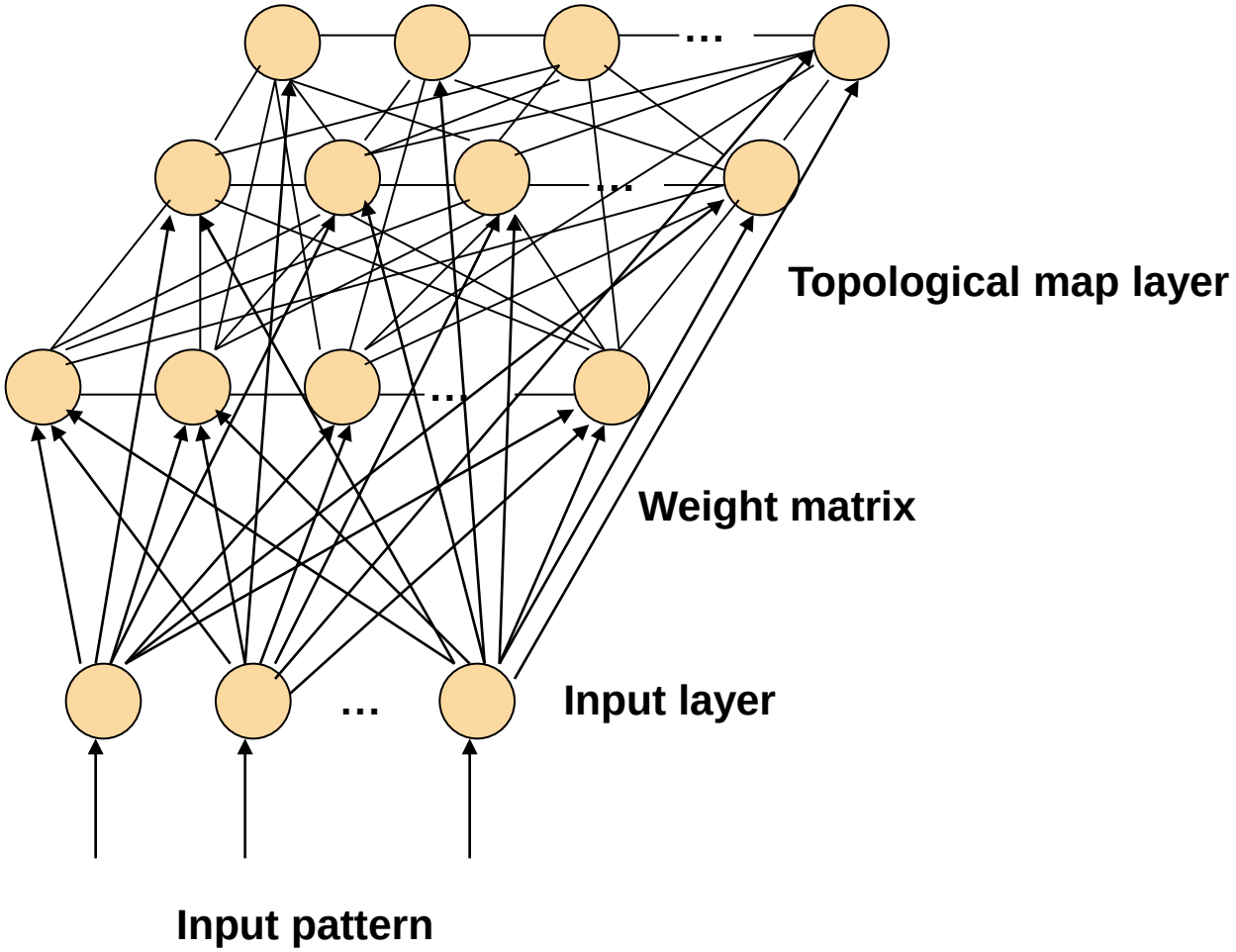
Hopfield Network characteristics

Neuron layers	One layer
Connection pattern	Fully connected feedback
Input value types	Discrete vectors
Net function Activation function	Weighted sum Step function
Learning method Learning rule	Non supervised Hebbian learning algorithm

Overview of Kohonen Self Organizing Maps

- A Self-Organizing Map (SOM) developed by Kohonen in the early 1980s (Kohonen, 2001) is an artificial neural network that constructs topology-preserving mappings of the training data.
- A SOM consists of the two layers: an input layer and a feature map (a topological configuration as an output layer)
- Network nodes become specifically tuned to various input signal patterns or classes of patterns in an orderly mode.
- Network learning process is competitive and unsupervised, and as a result of this competition only one network node at a time is activated corresponding to each input.
- A SOM can be seen as a neural network based divisive clustering approach (Kohonen, 2001).

Kohonen Self Organizing Map architecture



Self Organizing Map for pattern classification

When a SOM is used for the pattern classification, the input patterns are assigned to a series of clusters on the basis of the similarity between them and the cluster center/reference vectors. Before initiating the task of classification or pattern recognition, the user has to predefine a topology of nodes (typically a two dimensional rectangular or hexagonal grid, where each node into the grid represents a cluster). Initially, random reference vector are generated and assigned for each node. After that, an iterative process involves the following steps:

1. An input pattern is chosen at random, and the node that maps closest to it is selected,
2. The reference vector of the selected node (in input pattern space) is then adjusted so that it is more similar to the input pattern assigned,
3. The reference vectors of the others nodes that are close to the selected node on the two-dimensional grid are also adjusted.

Kohonen Self Organizing Map characteristics

Neuron layers	One input layer One topological map layer
Connection pattern	Feedforward Inhibitory connection among all nodes into the feature map
Input value types	Real, binary
Net function Activation function	Euclidean distance Sigmoid
Learning method Learning algorithm	Competitive Self-organizing algorithm

Applications of artificial neural networks

- Neural networks for pattern classification
- Neural networks for signals processing
- Neural networks for pattern association
- Neural networks for clustering
- Neural networks for image processing
- Neural networks for time series prediction
- Neural networks for control

Neural networks for pattern classification

Task	Neural network models	Application domain
Classification	Perceptron, Multilayer Perceptron Networks with Backpropagation Learning	• Pattern classification (for example, character recognition problems) • Complex logical operations • Speech analysis

Neural networks for signal processing

Task	Neural network models	Application domain
Signal processing as pattern classification	Multilayer Perceptron Networks with Backpropagation Learning	• Speech recognition • optical character recognition • bar code recognition • human face recognition

Neural networks for pattern association

Task	Neural network models	Application domain
Pattern association	Hopfield Networks, Multilayer Perceptron Networks with Backpropagation Learning	• Auto-association or associative memory task (for example, generation of an uncorrupted image from a corrupted version) • Hetero-association (for example, translating English word inputs into equivalent Spanish word outputs)

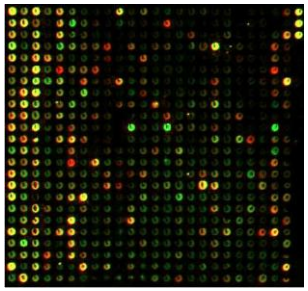
Neural networks for clustering

Task	Neural network models	Application domain
Clustering Vector quantization	Self Organizing Maps (SOM), Evolving Neural Networks, Neural Network Models with Unsupervised Learning	• Interpretation of gene expression patterns • Location determination in wireless environment

Neural networks for clustering

An Evolving Neural Network for the Interpretation of Gene Expression Patterns

Márquez, M.C., González, P.P. and Lagúnez, J. (2005). OMICS A Journal of Integrative Biology, Volume 9, Number 2, pages 209-217



Microarray

Conversion algorithm

$X_1 = (X_{11}, X_{12}, ..., X_{1m})$

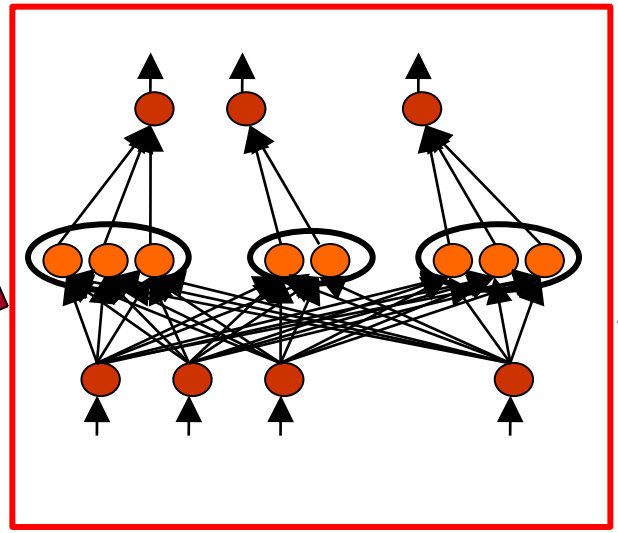
$X_2 = (X_{21}, X_{22}, ..., X_{2m})$

$X_3 = (X_{31}, X_{32}, ..., X_{3m})$

. . .

$X_n = (X_{n1}, X_{n2}, ..., X_{nm})$

Gene expression data



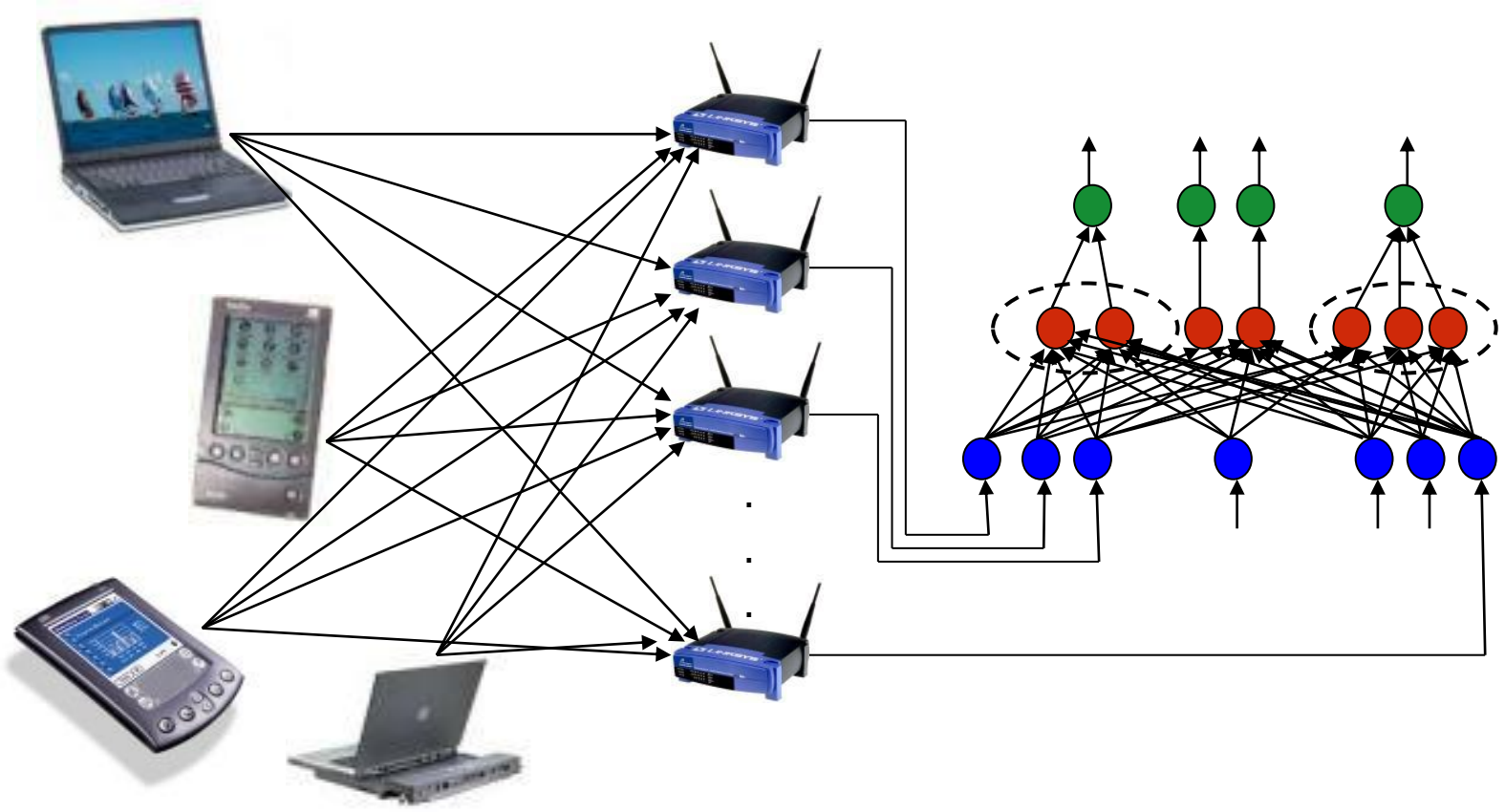
Association between gene expression patterns

Clustering gene expression patterns

Neural networks for clustering

A Neural Network for RF-based User Location in Wireless Systems

González, P.P., Gramellini, C., Buda, C., Lucchi, M. and Dardari, D.
DEIS, Università degli Studi di Bologna, http://wireless.deis-ce.unibo.it



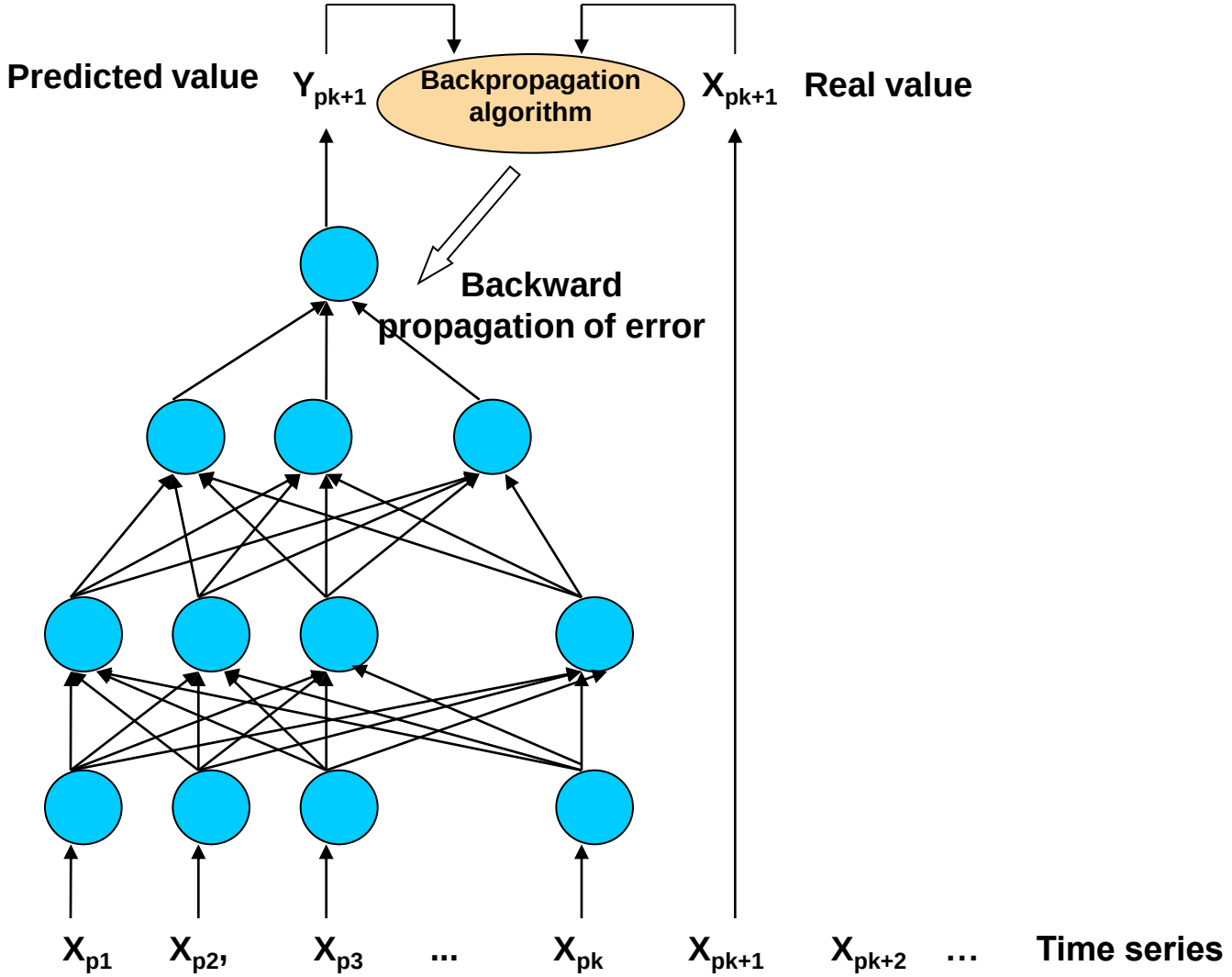
Neural networks for image processing

Task	Neural network models	Application domain
Image processing	Feedforward Networks with Backpropagation Algorithm (for pattern recognition) Unsupervised Neural Networks (for feature extraction)	• Biomedical image analysis (detection and characterization of disease patterns, quantification and segmentation, compression, restoration, etc.)

Neural networks for time series prediction

Task	Neural network models	Application domain
Time series prediction	Feedforward networks, Time Delay Networks and Recurrent Neural Networks	• Economic and financial forecasting • Medical risk prediction • Chaotic time series prediction

Neural networks for time series prediction



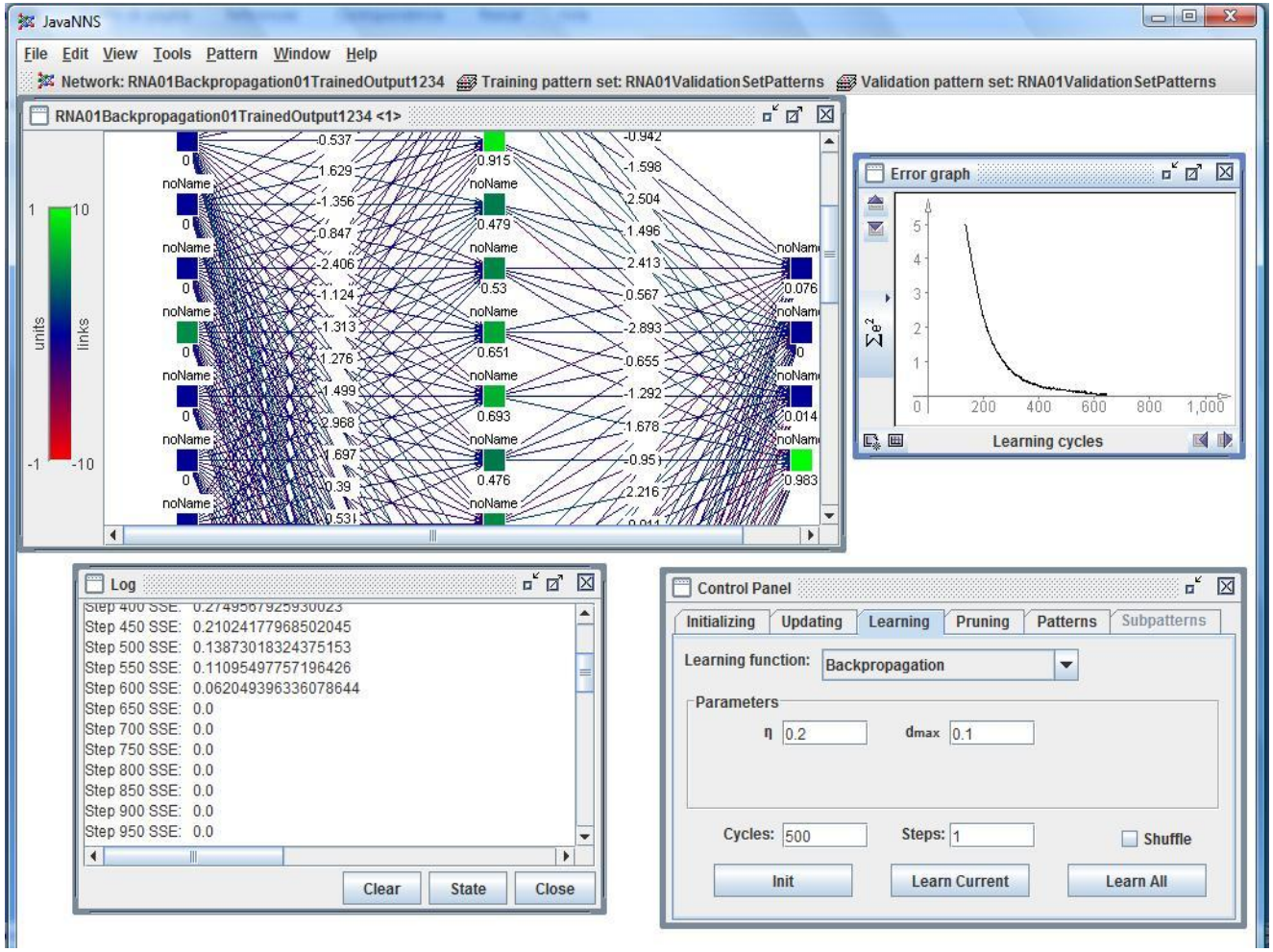
Neural networks for control

Task	Neural network models	Application domain
Control applications	Feedforward networks, Recurrent Neural Networks	• Manufacturing and industrial applications

Neural networks tools

Tool	Description	http
JavaNNS	Simulator for neural networks	http://www.ra.cs.uni-tuebingen.de/downloads/JavaNNS/
DemoGNG	A Java applet that implements several competitive learning networks	http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG/GNG.html

JavaNNS: a neural network simulator



http://www.ra.cs.uni-tuebingen.de/downloads/JavaNNS/

Leak detection in pipelines using a Backpropagation Neural Network

References

- Mehrotra, K., Mohan, C.K. y S. Ranka. (1997). Elements of Artificial Neural Networks. MIT Press.
- Fausett, L. V. (1994). Fundamentals of neural networks. Prentice Hall.
- Arbib, M. (1998). The handbook of the brain theory and neural networks. MIT Press.
- Hertz, J., Krogh, A. y Palmer, R.G. (1991). Introduction to the theory of neural computation. Addison-Wesley.