Artificial Neural Network (ANN)

A. Introduction to neural networks
B. ANN architectures
- Feedforward networks
- Feedback networks
- Lateral networks

C. Learning methods
- Supervised learning
- Unsupervised learning
- Reinforced learning

D. Learning rule on supervised learning
- Gradient descent,
- Widrow-hoff (LMS)
- Generalized delta
- Error-correction

E. Feedforward neural network with Gradient descent optimization

Introduction to neural networks

Definition: the ability to learn, memorize and still generalize, prompted research in algorithmic ***modeling of biological neural systems***

Do you think that computer smarter than human brain?

"While successes have been achieved in modeling biological neural systems, there are still no solutions to the complex problem of modeling intuition, consciousness and emotion - which form **integral parts of human intelligence"…(**Alan Turing, 1950)

---**Human brain** has the ability to perform tasks such as pattern recognition, perception and motor control much faster than any computer---

Facts of Human Brain

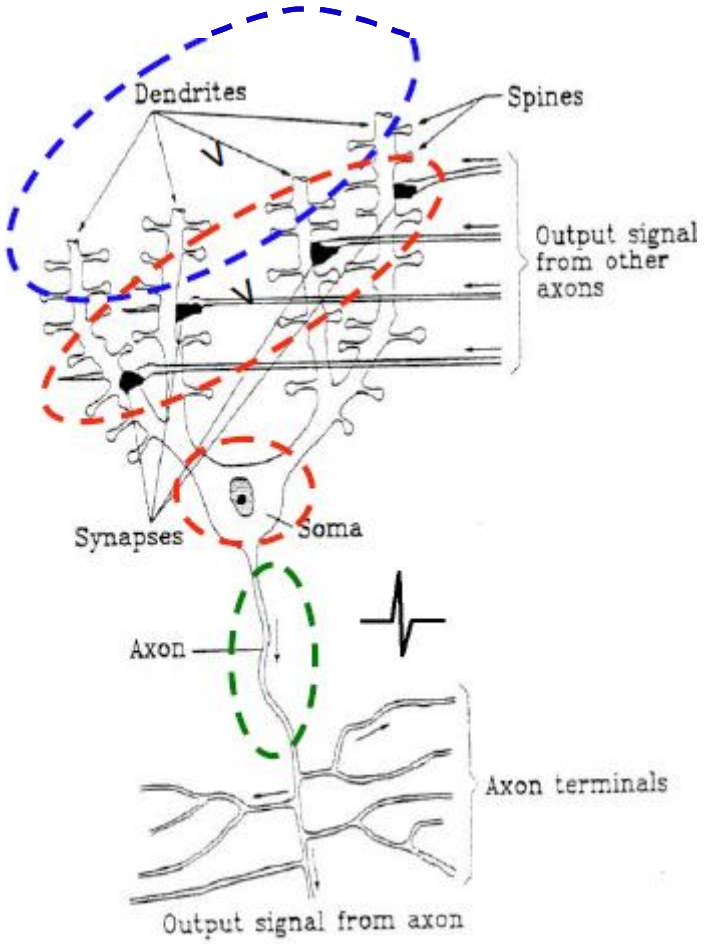
(complex, nonlinear and parallel computer)

- The brain contains about 10^{10} (100 billion) basic units called neurons
- Each neuron connected to about 10^4 other neurons
- Weight: birth 0.3 kg, adult ~1.5 kg
- Power consumption 20-40W (~20% of body consumption)
- Signal propagation speed inside the axon ~90m/s in ~170,000 Km of axon length for adult male
- Firing frequency of a neuron ~250 – 2000Hz
- Operating temperature: 37±2°C
- Sleep requirement: average 7.5 hours (adult)

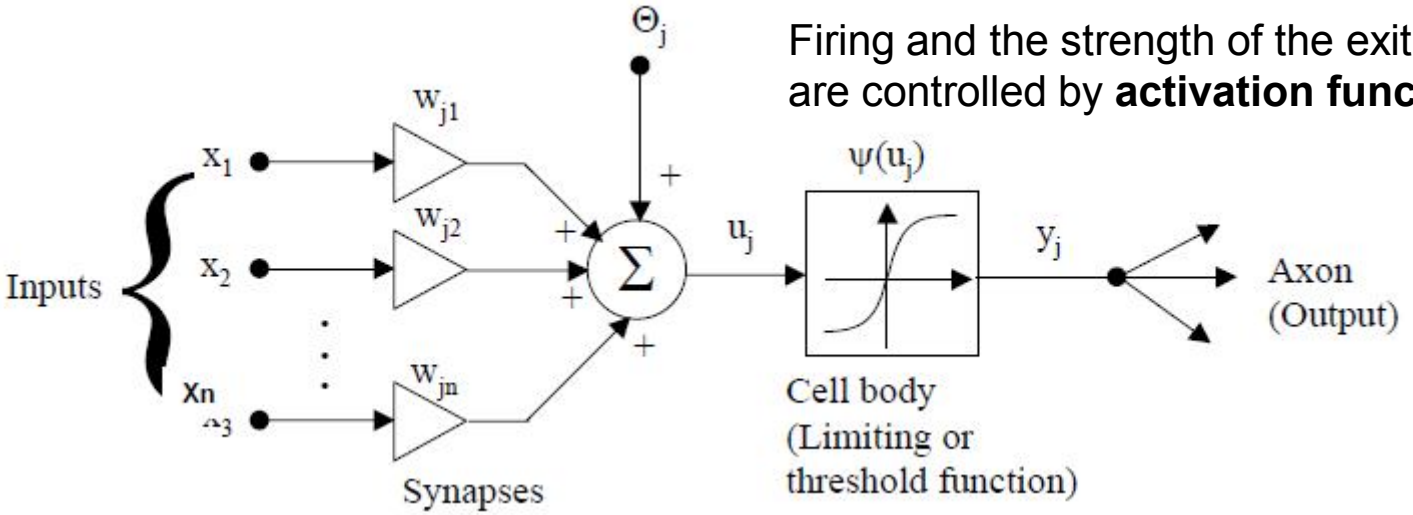
Intel Pentium 4 1.5GHz	
Number of transistors	4.2×10^7
Power consumption	up to 55 Watts
Weight	0.1 kg cartridge w/o fans, 0.3 kg with fan/heatsink
Maximum firing frequency	1.5 GHz
Normal operating temperature	15-85°C
Sleep requirement	0 (if not overheated/ overclocked)
Processing of complex stimuli	if can be done, takes a long time

Biological neuron

- *Soma*: Nucleus of neuron (the cell body) - process the input
- *Dendrites*: long irregularly shaped filaments attached to the soma – input channels
- *Axon*: another type link attached to the soma – output channels
- Output of the axon: voltage pulse (spike) that lasts for a ms
- Firing of neuron – membrane potential
- Axon terminates in a specialized contact called the **synaptic junction** – the electrochemical contact between neurons
- The size of synapses are believed to be linked with ***learning***
- Larger area: excitatory—smaller area: inhibitory



Artificial neuron model (McCulloh-Pitts model, 1949)



Firing and the strength of the exiting signal are controlled by **activation function (AF)**

Types of AF:
- *Linear*
- *Step*
- *Ramp*
- *Sigmoid*
- *Hyperbolic tangent*
- *Gaussian*

Θ_j : external threshold, offset or bias
w_{ji} : synaptic weights
x_i : input
y_j : output

$$y_j = \psi\left(\sum_{i=1}^{n} w_{ji}x_i + \Theta_j\right)$$

…..Another model-**Product unit**

Allow higher-order combinations of inputs, having the advantage of increased information capacity

Different NN types

- Single-layer NNs, such as the Hopfield network
- **Multilayer feedforward NNs**, for example standard backpropagation, functional link and product unit networks
- **Temporal NNs**, such as the Elman and Jordan simple recurrent networks as well as time-delay neural networks
- Self-organizing NNs, such as the Kohonen self-organizing feature maps and the learning vector quantizer
- Combined feedforward and self-organizing NNs, such as the **radial basis function networks**

The ANN applications

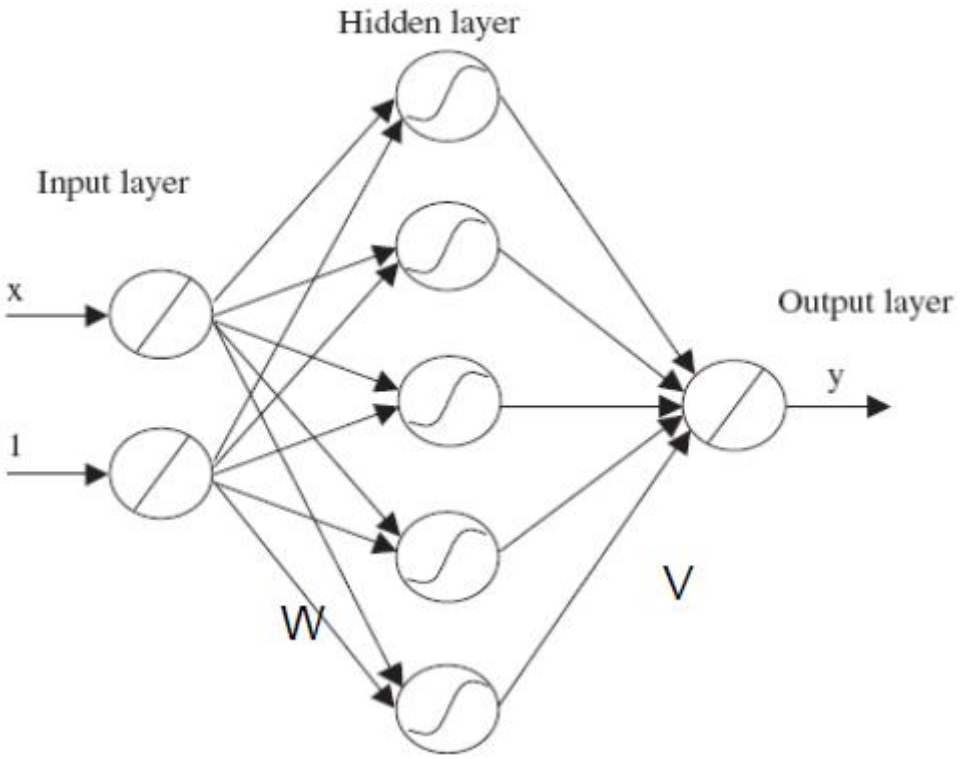
- ***Classification,*** *the aim is to predict the class of an input vector*
- ***Pattern matching****, the aim is to produce a pattern best associated with* a given input vector
- ***Pattern completion****, the aim is to complete the missing parts of a given* input vector
- ***Optimization****, the aim is to find the optimal values of parameters in an* optimization problem
- ***Control****, an appropriate action is suggested based on given an input vectors*
- ***Function approximation/times series modeling****, the aim is to learn the* functional relationships between input and desired output vectors;
- ***Data mining****, with the aim of discovering hidden patterns from data* (knowledge discovery)

ANN architectures

- Neural Networks are known to be universal function approximators
- Various architectures are available to approximate any nonlinear function
- Different architectures allow for generation of functions of different complexity and power

- *Feedforward networks*
- *Feedback networks*
- *Lateral networks*

Feedforward Networks



Network size: n x m x r = 2x5x1
W_{mn}: input weight matrix
V_{rm}: output weight matrix

- *No feedback within the network*
- *The coupling takes place from one layer to the next*
- *The information flows, in general, in the forward direction*

Input layer: Number of neurons in this layer corresponds to the number of inputs to the neuronal network. This layer consists of **passive nodes**, i.e., which do not take part in the actual signal modification, but only transmits the signal to the following layer.

- **Hidden layer:** This layer has arbitrary number of layers with arbitrary number of neurons. The nodes in this layer take part in the signal modification, hence, they are **active**.

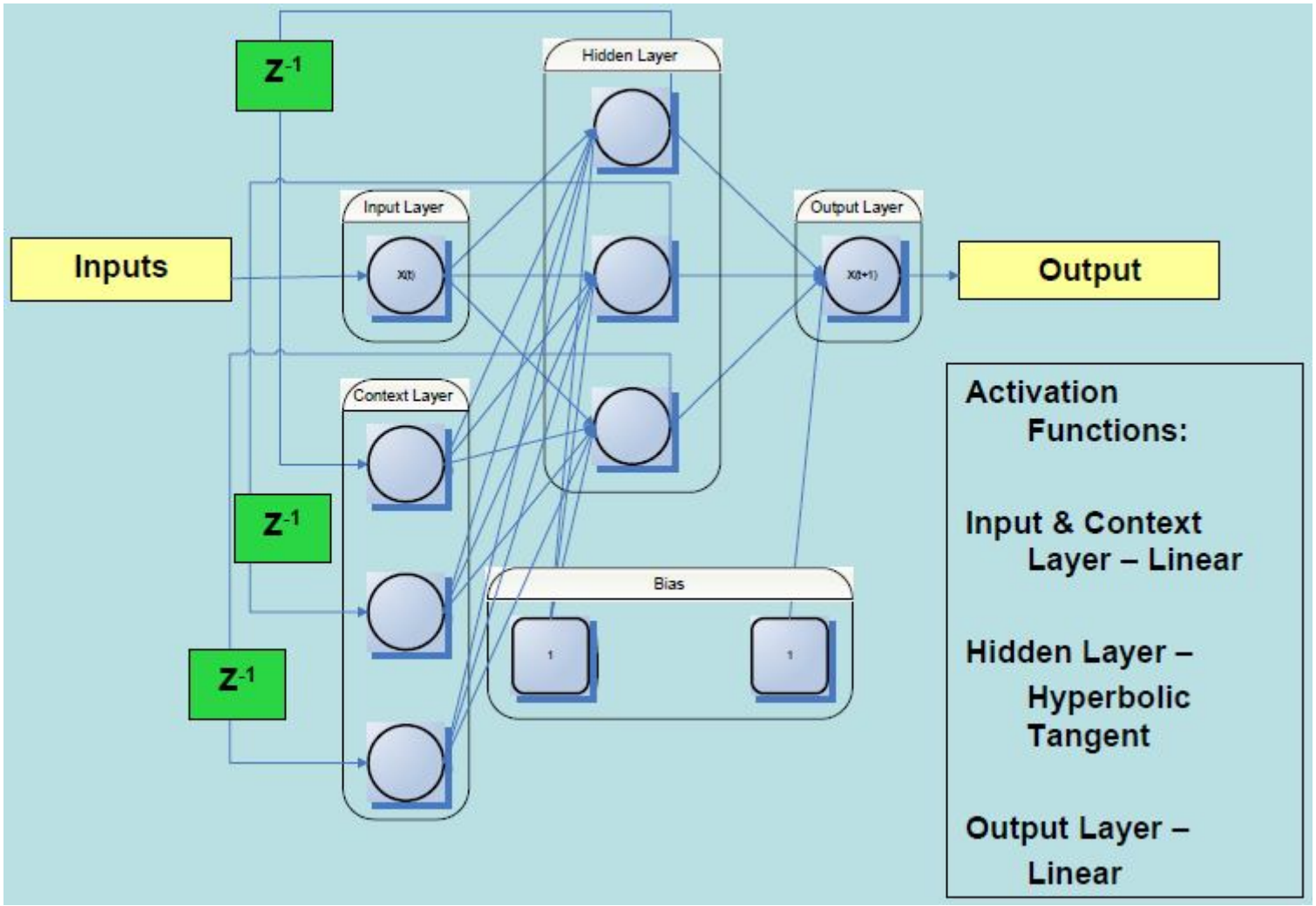
- **Output layer:** The number of neurons in the output layer corresponds to the number of the output values of the neural network. The nodes in this layer are **active** ones.

FFNN can **have more than one hidden layer.** *However*, it has been proved that FFNNs with one hidden layer has enough to approximate any continuous function [Hornik 1989].

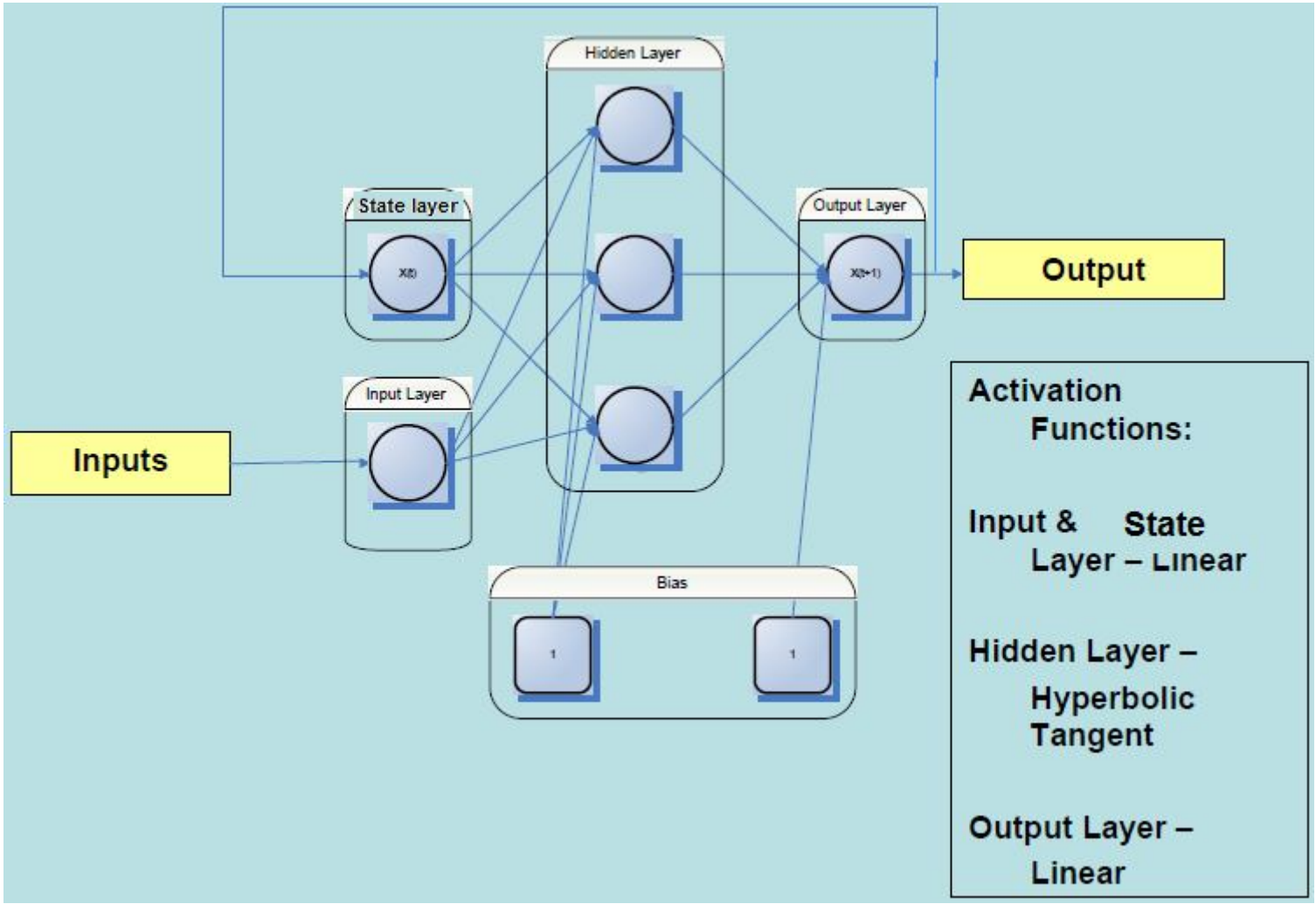
Feedback networks

Elman Recurrent Network

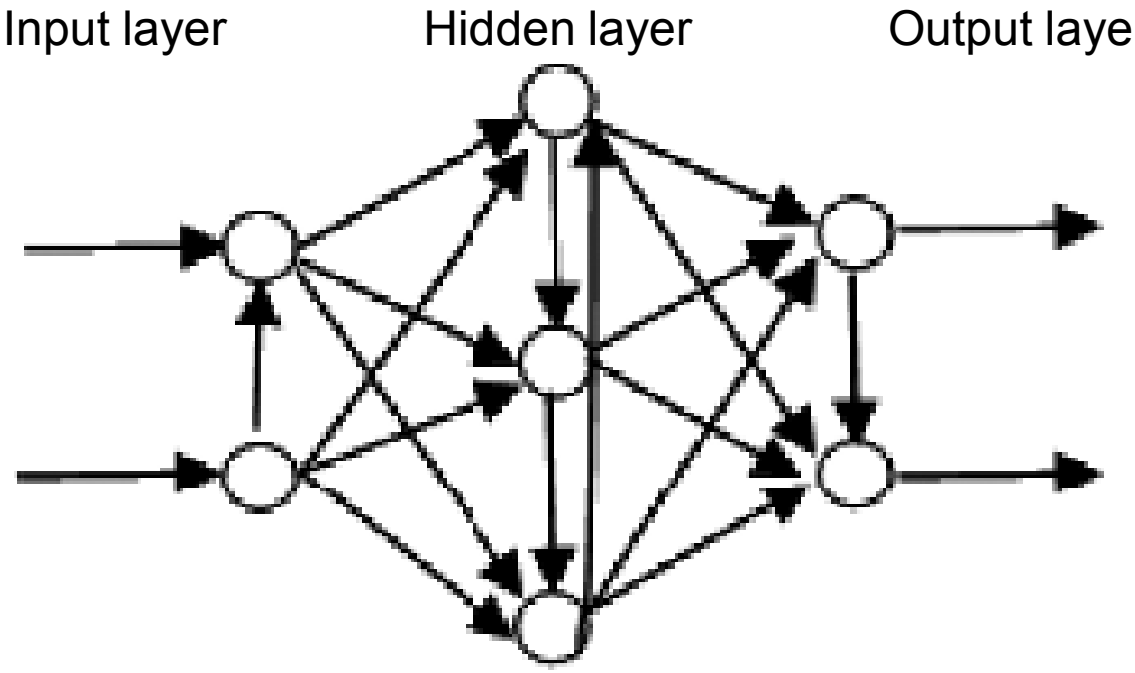
The output of a neuron is either directly or indirectly fed back to its input via other linked neurons→ used in complex pattern recognition tasks, e.g., speech recognition etc.



Feedback networks
Jordan Recurrent Network



Lateral Networks



- There exist couplings of neurons within one layer
- There is no essentially explicit feedback path amongst the different layers
- This can be thought of as a **compromise between the forward and feedback network**

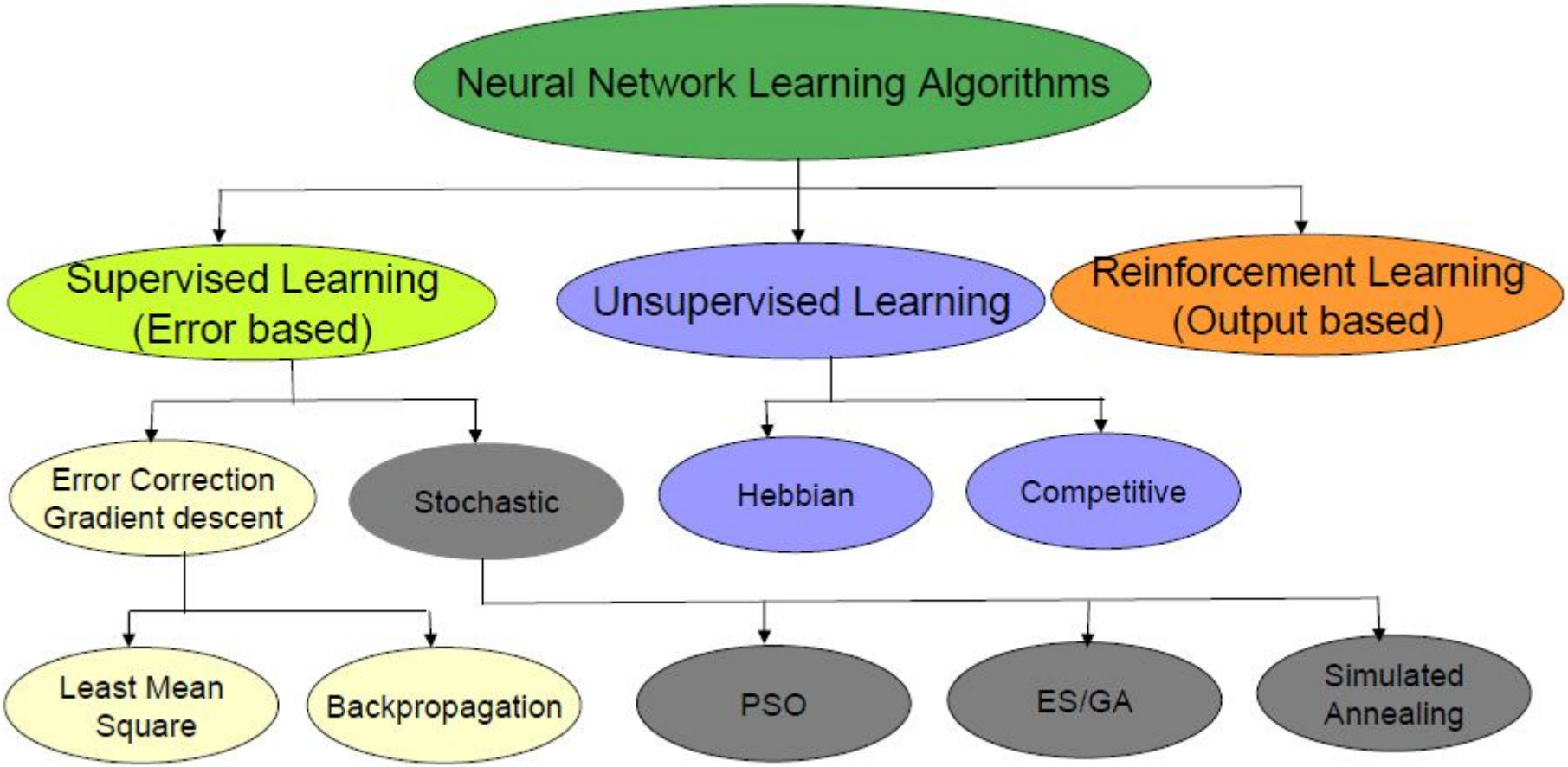
Learning methods

- Artificial neural networks work through the optimized weight values.
- The method by which the optimized weight values are attained is called ***learning***
- In the learning process → try to teach the network how to produce the output when the corresponding input is presented
- When learning is complete: the trained neural network, with the updated optimal weights, should be able to produce the output within desired accuracy corresponding to an input pattern.

Learning methods

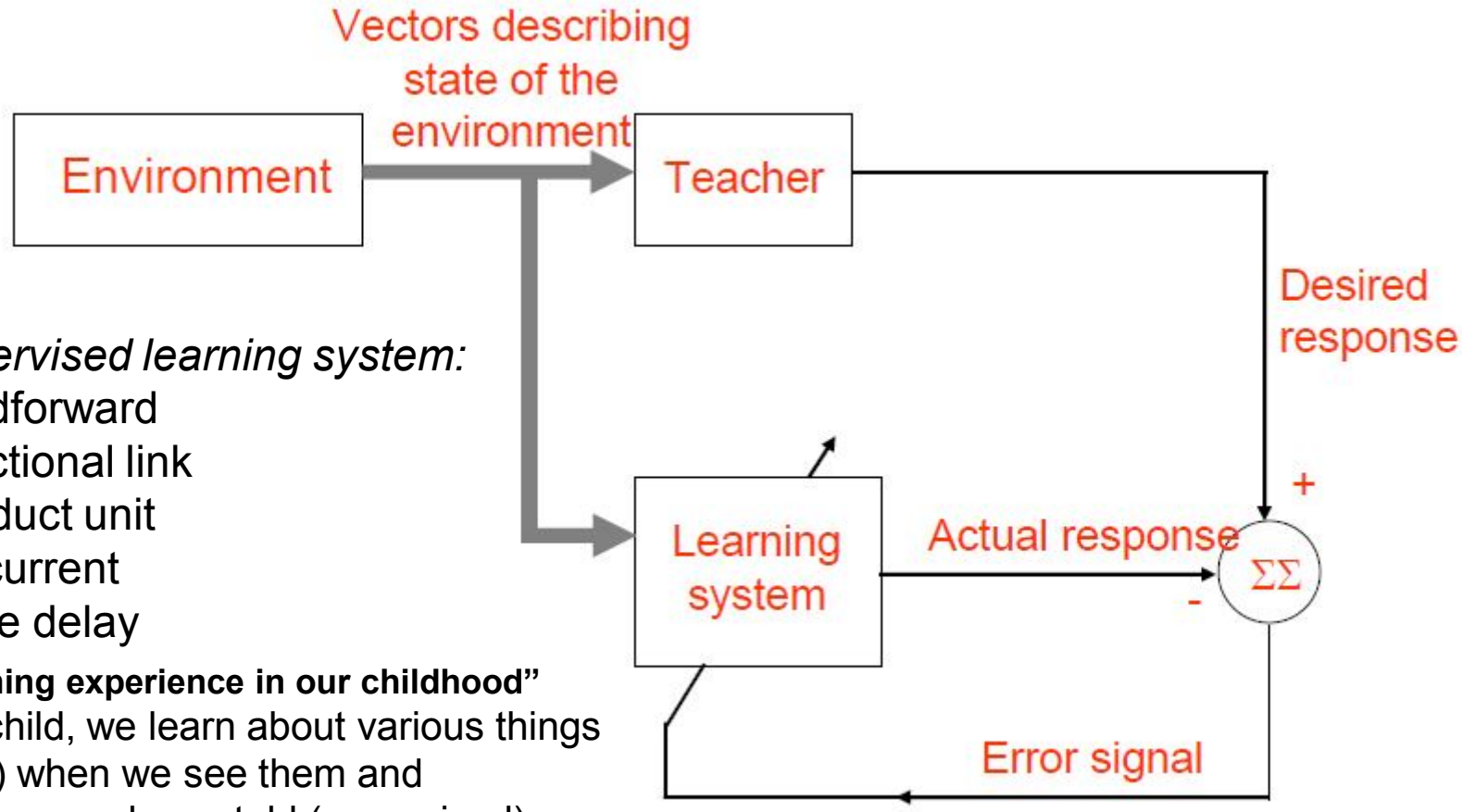
- Supervised learning
- Unsupervised learning
- Reinforced learning

Classification of Learning Algorithms



Supervised learning

Supervised learning means guided learning by "teacher"; requires a training set which consists of input vectors and a target vector associated with each input vector



Supervised learning system:

- feedforward
- functional link
- product unit
- Recurrent
- Time delay

"Learning experience in our childhood"

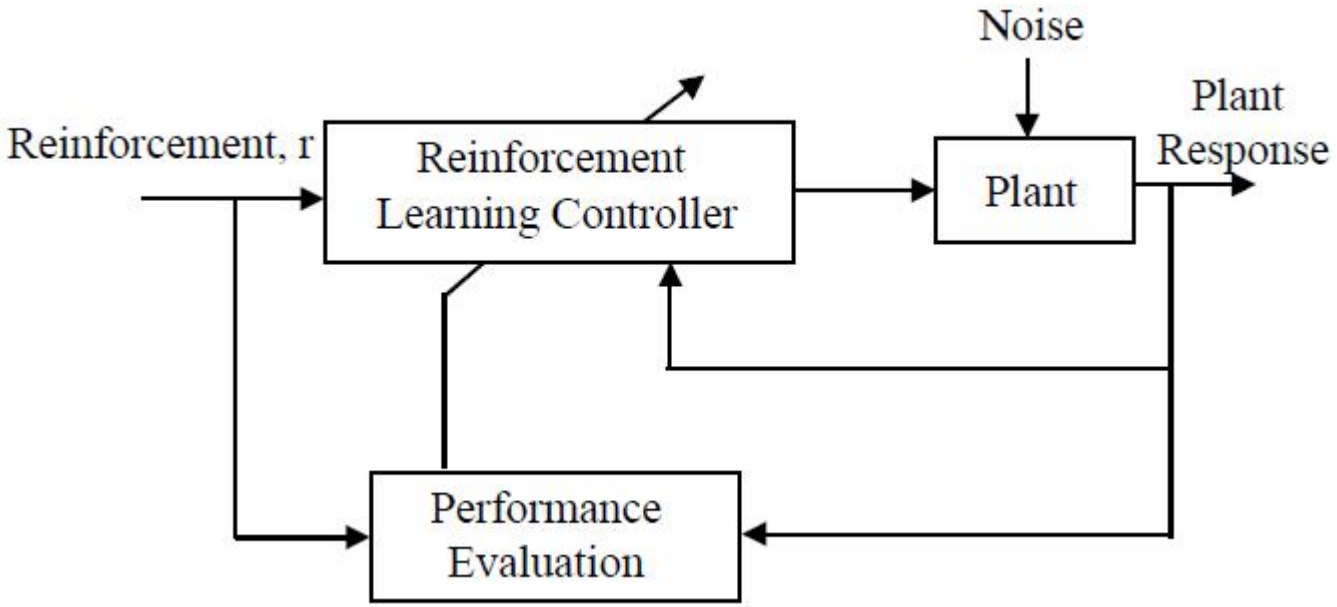
As a child, we learn about various things (input) when we see them and simultaneously are told (supervised) about their names and the respective functionalities (desired response).

Unsupervised learning

- The objective of unsupervised learning is to discover patterns or features in the input data with no help from a teacher, basically performing a clustering of input space.
- The system learns about the pattern from the data itself without *a* ***priori knowledge***. *This is similar to our learning experience in adulthood*

  "For example, often in our working environment we are thrown into a project or situation which we know very little about. However, we try to familiarize with the situation as quickly as possible using our previous experiences, education, willingness and similar other factors"
- Hebb's rule: It helps the neural network or neuron assemblies to remember specific patterns much like the **memory**. From that stored knowledge, similar sort of incomplete or spatial patterns could be recognized. This is even faster than the delta rule or the backpropagation algorithm because there is no repetitive presentation and training of input–output pairs.

Reinforced learning

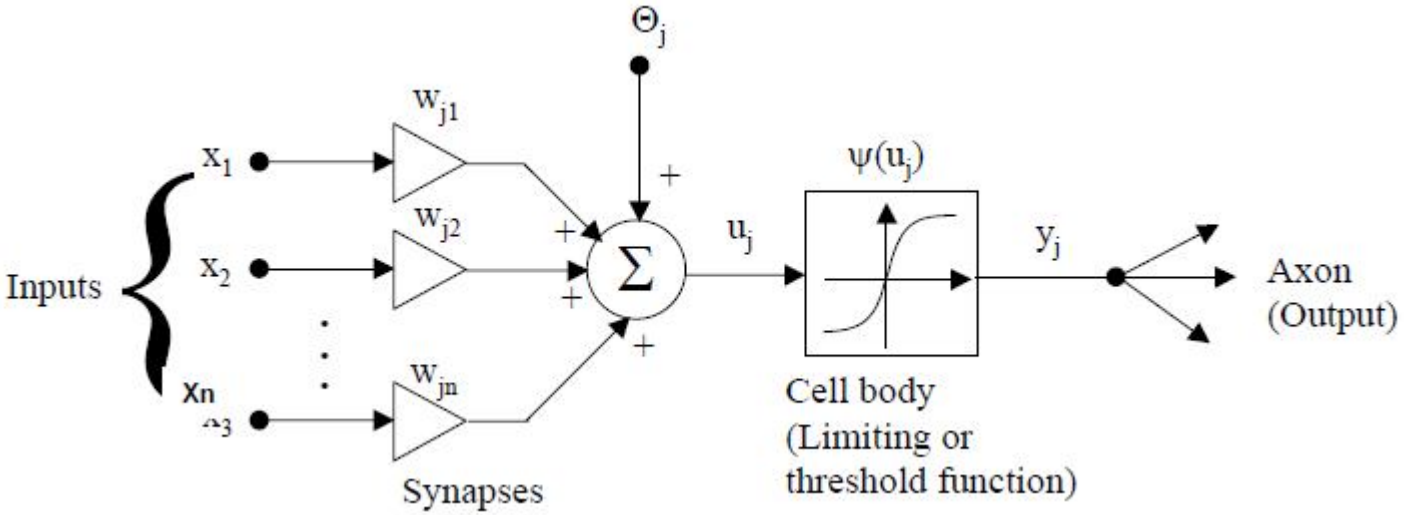


•A ‘teacher’ though available, does not present the expected answer but only indicates if the computed output is correct or incorrect
•The information provided helps the network in its learning process
•A *reward is given for a correct answer* computed and *a penalty for a wrong answer*

Leaning algorithm in Supervised learning

- Gradient descent
- Widrow-hoff (LMS)
- Generalized delta
- Error-correction

Single neuron



Gradient Descent

- **Gradient descent (GD)…(**not the first but used most)
- GD is aimed to find the weight values that minimize *Error*
- GD requires the definition of an error (or objective) function to measure the neuron's error in approximating the target

$$f_j = \psi\left(\sum_{i=1}^{n+1} w_{ji} x_i\right) \qquad Error = \sum_{p=1}^{P} (t_p - f_p)^2$$

Where t_p and f_p are respectively the target and actual output for patterns p

The updated weights:

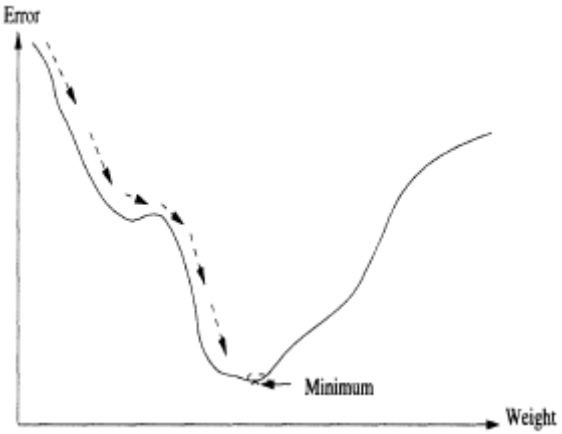
$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

$$with \quad \Delta w_i(t) = \eta(-\frac{\partial E}{\partial w_i})$$

$$where \quad \frac{\partial E}{\partial w_i} = -2(t_p - f_p)\frac{\partial f}{\partial u_p} x_{i.p}$$

where *η :learning rate*
w_i(t+1):new weights

Analogy: Suppose we want to come down (descend) from a high hill (higher error) to a low valley (lower error). We move along the negative gradient or slopes. By doing so, we take the *steepest path to* the downhill valley→ steepest descent algorithm



The calculation of the partial derivative of *f with respect to u_p (the net* input for pattern *p) presents a problem for all discontinuous activation functions,*
such as the **step and ramp functions**

Widrow-Hoff learning rule

Widrow-hoff Least-Means-Square (LMS)

Assume that $f = u_p$

The weights are updated using:

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

$$with \quad \Delta w_i(t) = \eta(-\frac{\partial E}{\partial w_i})$$

$$where \quad \frac{\partial E}{\partial w_i} = -2(t_p - f_p)\frac{\partial f}{\partial u_p}x_{i,p} = -2(t_p - f_p).1.x_{i,p} = -2(t_p - f_p)x_{i,p}$$

$$\boxed{w_i(t+1) = w_i(t) + 2\eta(t_p - f_p)x_{i,p}}$$

One of the first algorithms used to train multiple adaptive linear neurons (Madaline) [Widrow 1987, Widrow and Lehr 1990]

Generalized delta

Assume: **differentiable activation functions; such as sigmoid function**
The weights are updated using:

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

$$with \quad \Delta w_i(t) = \eta(-\frac{\partial E}{\partial w_i})$$

$$where \quad \frac{\partial E}{\partial w_i} = -2(t_p - f_p)\frac{\partial f}{\partial u_p}x_{i,p} = -2(t_p - f_p).f_p(1-f_p).x_{i,p}$$

$$\boxed{w_i(t+1) = w_i(t) + 2\eta(t_p - f_p).f_p(1-f_p).x_{i,p}}$$

Error-correction

Assume that *binary-valued functions are used, e.g* ***the step function***.

The weights are updated using:

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

$$with \quad \Delta w_i(t) = \eta(-\frac{\partial E}{\partial w_i})$$

$$where \quad \frac{\partial E}{\partial w_i} = -2(t_p - f_p)x_{i,p}$$

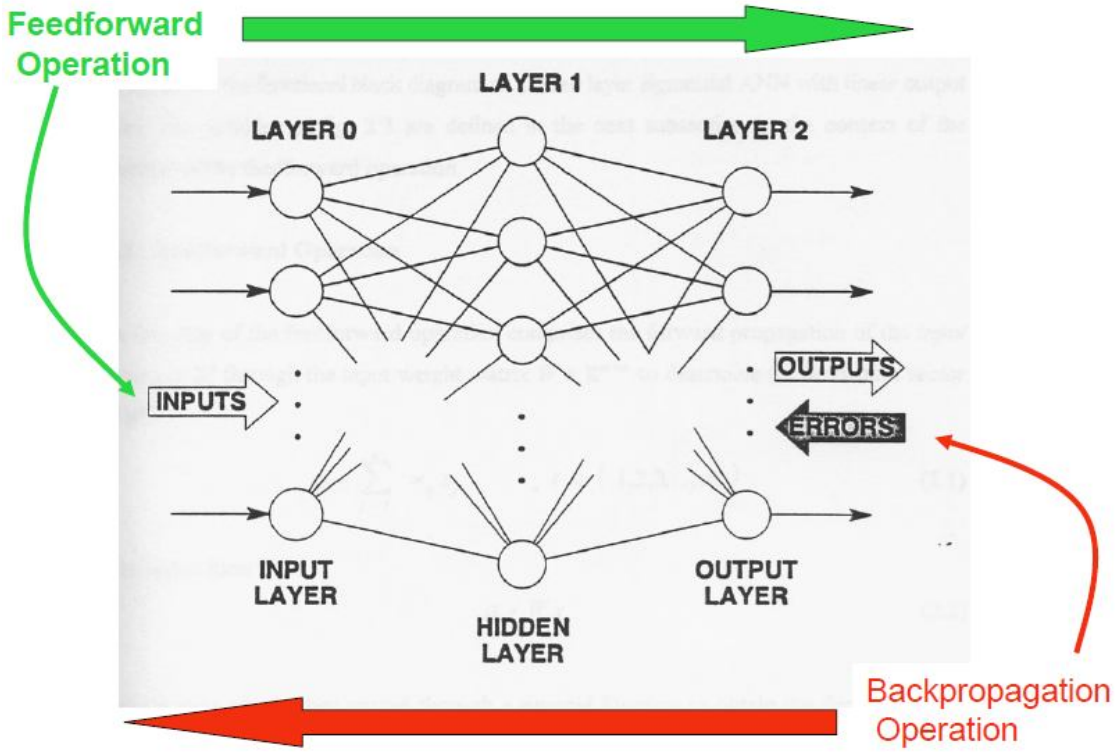
$$\boxed{w_i(t+1) = w_i(t) + 2\eta(t_p - f_p)x_{i,p}}$$

Weights are only adjusted when the neuron responds in error

$$where \quad (t_p - f_p) = 1 \quad or \quad (t_p - f_p) = -1$$

Feedforward neural network with Gradient descent optimization

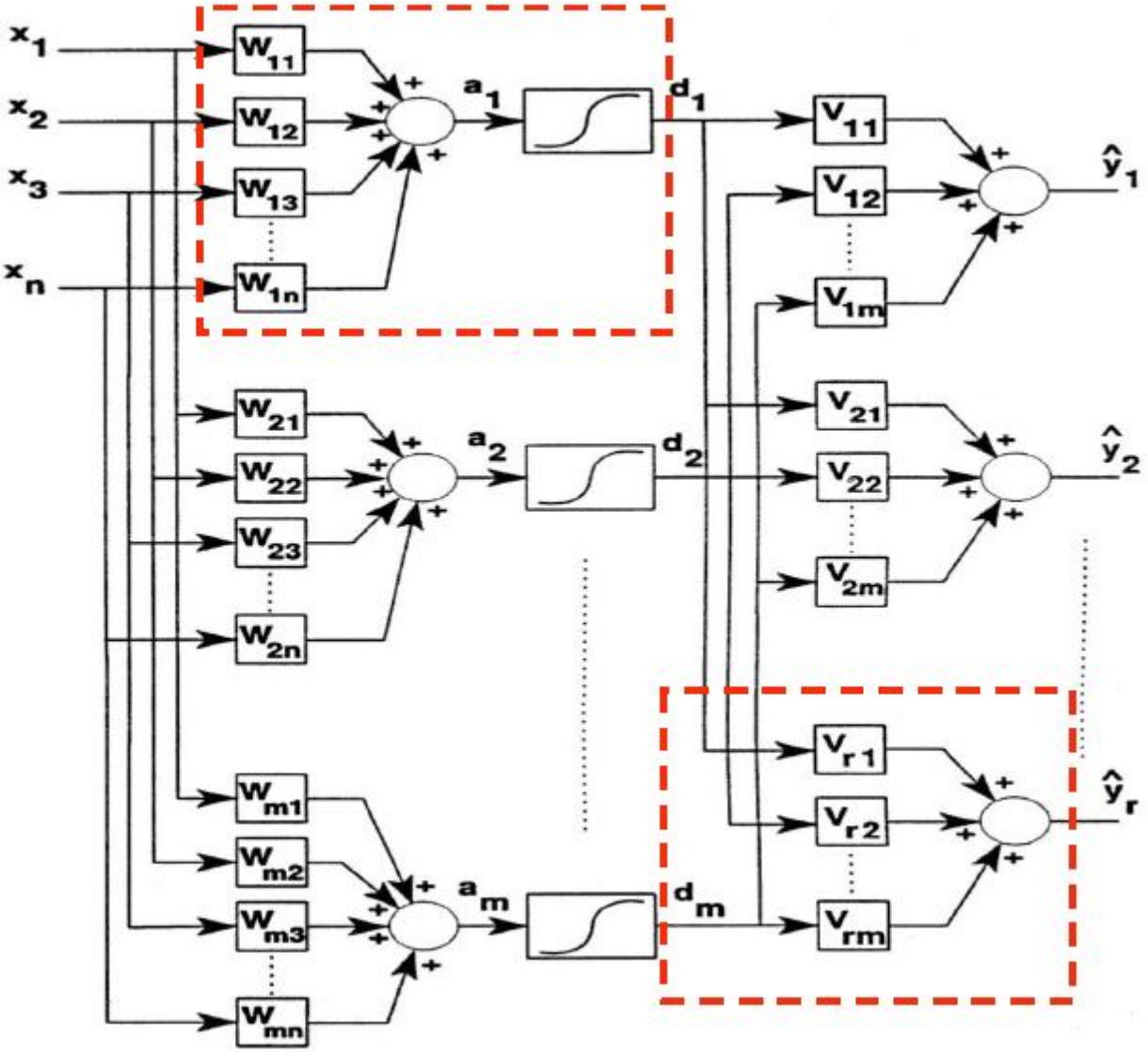
Input vectors → actual value is calculated then error is calculated



The error gradient respect to network's weight is calculated by propagating the error backward through network
Once the error gradient is calculated, the weight is adjusted

More details…

Functional Diagram of FFNN



Feedforward Operation

Input vector x_j *where j =1 to n (number of inputs)*
Input weight matrix W_{ij} *where i = 1 to m (hidden neurons)*

Step 1: *Activation vector* a_i :

$$a_i = \sum_{j=1}^{n} w_{ij} x_j \quad \rightarrow \quad \bar{a} = W\bar{x}$$

Decision vector d_i :

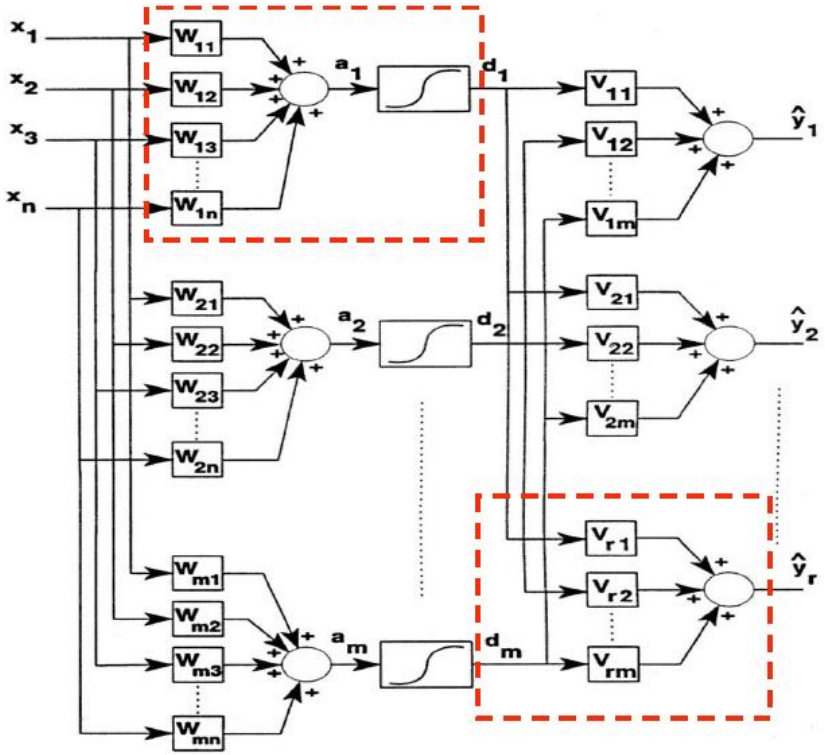
$$d_i = sig(a_i)$$

$$where \quad sig(.) = \frac{1}{1+e^{-(.)}}$$

Step 2: *Output vector* y_i *is given by (r is no. of* outputs):

$$y_h = \sum_{i=1}^{m} v_{hi} d_i \quad ; \quad h \in \{1,2,3,...r\}$$

$$\bar{\hat{y}} = V\bar{d}$$



Backpropagation Operation

The backpropagation training algorithm is based on *the principle of gradient descent* and is given as **half the square of the Euclidean norm** of the output error vector.

Step 1: The *output error* vector:

$$\bar{e}_y = \bar{y} - \hat{\bar{y}}$$

$$E(x, W, V, y) = \frac{1}{2}\left|\bar{e}_y\right|^2$$

Step 2: The *decision error* vector:

$$\bar{e}_d = V^T \bar{e}_y$$

"This is the objective function for NN learning that need to be **optimized** by the optimization methods"

The *activation error* vector:

$$\bar{e}_{ai} = d_i\,(1 - d_i)\,\bar{e}_{di}$$

Step 3: The *weights changes*:

$$\Delta V(k) = \gamma_g \bar{e}_y(k)\bar{d}^T(k) + \gamma_m \Delta V(k-1)$$

$$\Delta W(k) - \gamma_g \bar{e}_a(k)\bar{x}^T(k) + \gamma_m \Delta W(k-1)$$

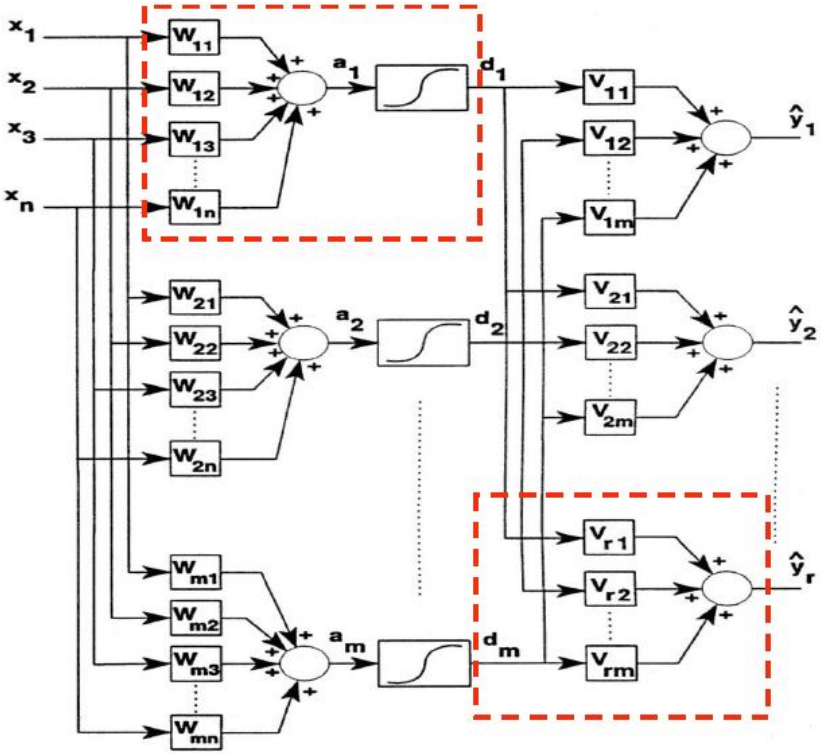
γ_g *and* γ_m *are learning and momentum rates, respectively*

The *weights updates:*

$$V(k+1) = V(k) + \Delta V(k)$$

$$W(k+1) = W(k) + \Delta W(k)$$

One set of weight modifications is called an ***epoch****, and many of these may be* required before the desired accuracy of approximation is reached.



Optimization methods to carry out NN learning

- **Local optimization**, where the algorithm ends up in a local optimum without finding a global optimum. *Gradient descent* and *scaled conjugate gradient* are local optimizers.
- **Global optimization,** where the algorithm searches for the global optimum by with mechanisms that allow greater search space explorations. Global optimizers include *Leapfrog, simulated annealing, evolutionary computing and swarm optimization*.

"Local and global optimization techniques can be combined to form hybrid training algorithms"

Weight Adjustments/Updates

Two types of supervised learning algorithms exist, based on when/how weights are updated:

- **Stochastic/Delta/(online) learning**, where the NN weights are adjusted after each pattern presentation. In this case the next input pattern is selected randomly from the training set, to prevent any bias that may occur due to the sequences in which patterns occur in the training set.
- **Batch/(offline) learning**, where the NN weight changes are accumulated and used to adjust weights only after all training patterns have been presented

Feedforward Neural Networks
(effects of weight variations)

$$\hat{y} = v_{11}\,\mathrm{sig}(w_{12}x + w_{11}) + v_{12}$$

$$\mathrm{sig}(.) = \frac{1}{1 + e^{(.)}}$$