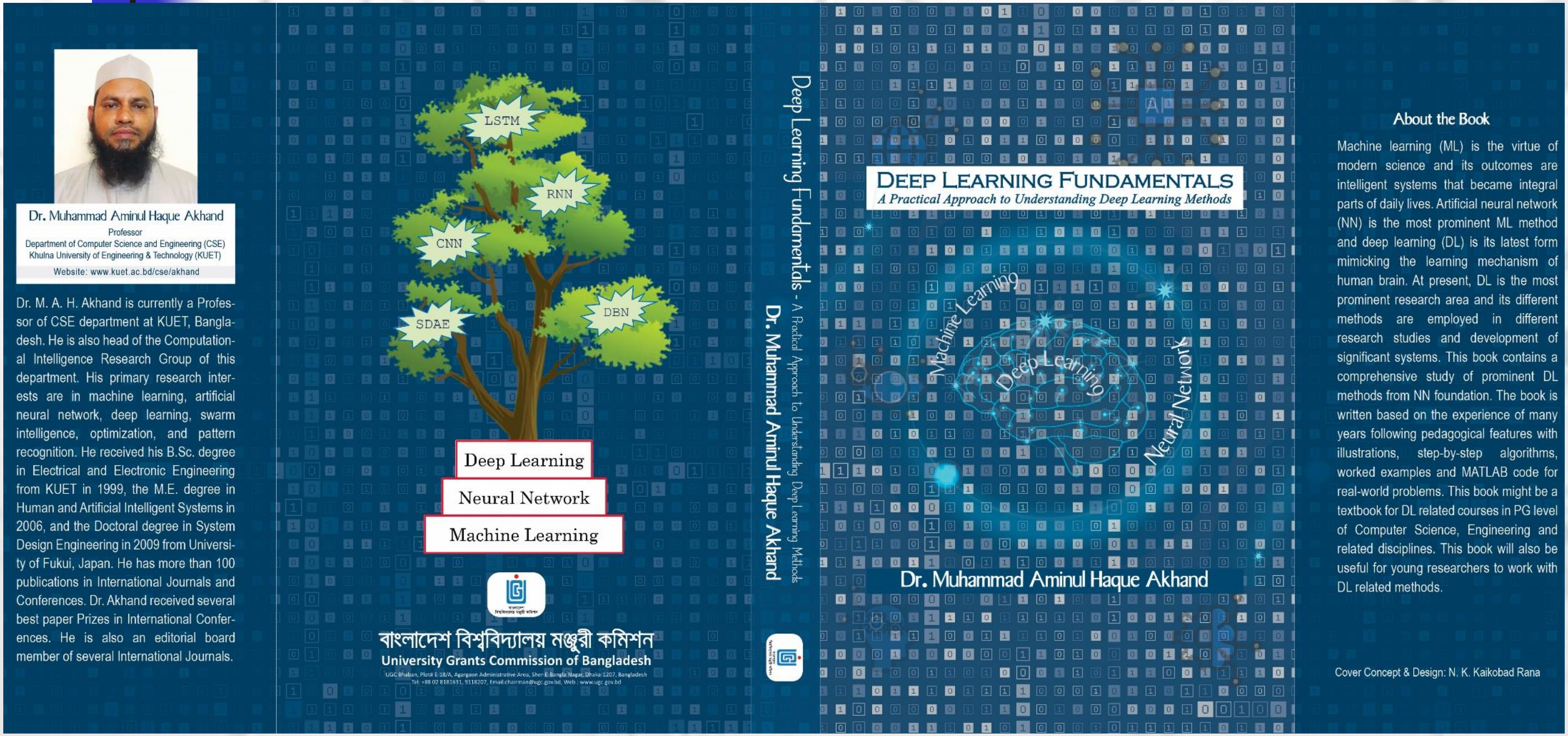
Deep Learning Fundamentals - A Practical Approach to Understanding Deep Learning Methods
ISBN:978-984-35-0812-6, University Grants Commission (UGC), Bangladesh, 2021

Chapter1 – Preliminaries

Biological Neuron to Artificial Neural Network

Dr. Md. Aminul Haque Akhand
Dept. of CSE, KUET

Deep Learning Fundamentals
- A Practical Approach to Understanding Deep Learning Methods



https://kuet.ac.bd/cse/akhand/

Contents

Why Artificial Neural Networks?

Human brain has many desirable characteristics not presented in modern computers:

massive parallelism,
distributed representation and computation,
learning ability,
generalization ability,
adaptability,
inherent contextual information processing,
fault tolerance, and
low energy consumption.

It is hope that devices based on biological neural networks will process some of these characteristics.

Why Artificial Neural Networks?

- Modern digital computers outperform humans in the domain of numeric computation and the related symbolic manipulation.

- However, humans can solve complex perceptual problems effortlessly, and more efficiently than the world's fastest computer.

- Human performs these remarkable benefits due to the characteristics of his/her neural system whose (neural system) working procedure is completely different from a conventional computer system.

Comparison of Brains and Traditional Computers

- 200 billion neurons, 32 trillion synapses
- Element size: 10^{-6} m
- Energy use: 25W
- Processing speed: 100 Hz
- Parallel, Distributed
- Fault Tolerant
- Learns: Yes
- Intelligent/Conscious: Usually

- 1 billion bytes RAM but trillions of bytes on disk
- Element size: 10^{-9} m
- Energy watt: 30-90W (CPU)
- Processing speed: 10^{9} Hz
- Serial, Centralized
- Generally not Fault Tolerant
- Learns: Some
- Intelligent/Conscious: Generally No

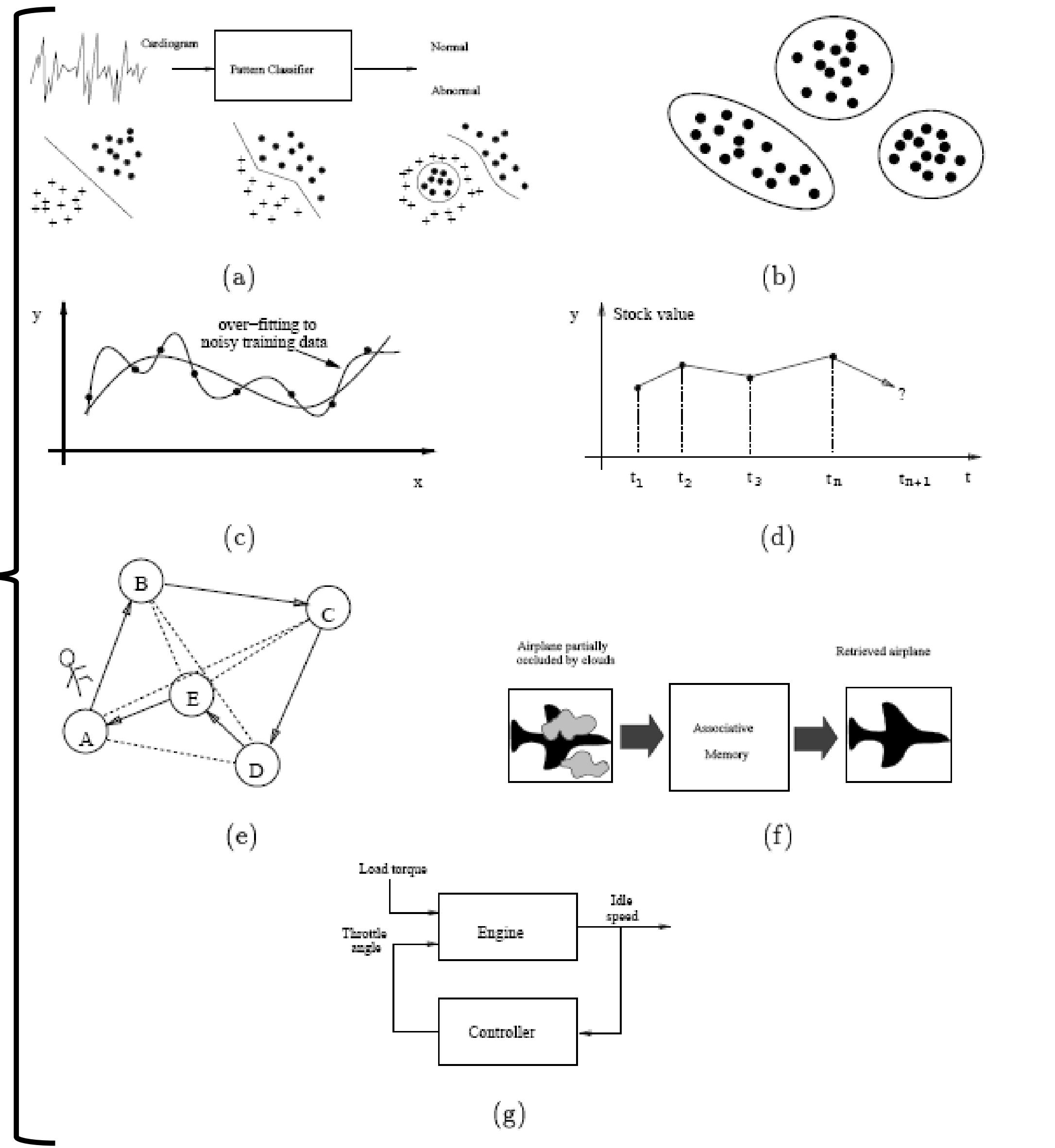
von Neumann Computer vs. Biological Neural System

	von Neumann Computer	Biological Neural System
processor	complex high-speed one (or a few)	simple low-speed a large number
memory	separate localized noncontent addressable	integrated in the neuron distributed content addressable
computing	centralized sequential strored-programs	distributed parallel self-learning
reliability	vulnerable	robust
expertise	numerical symbolic	perceptual problems manipulations
environment	well-defined	poorly defined unconstrained

The Goal

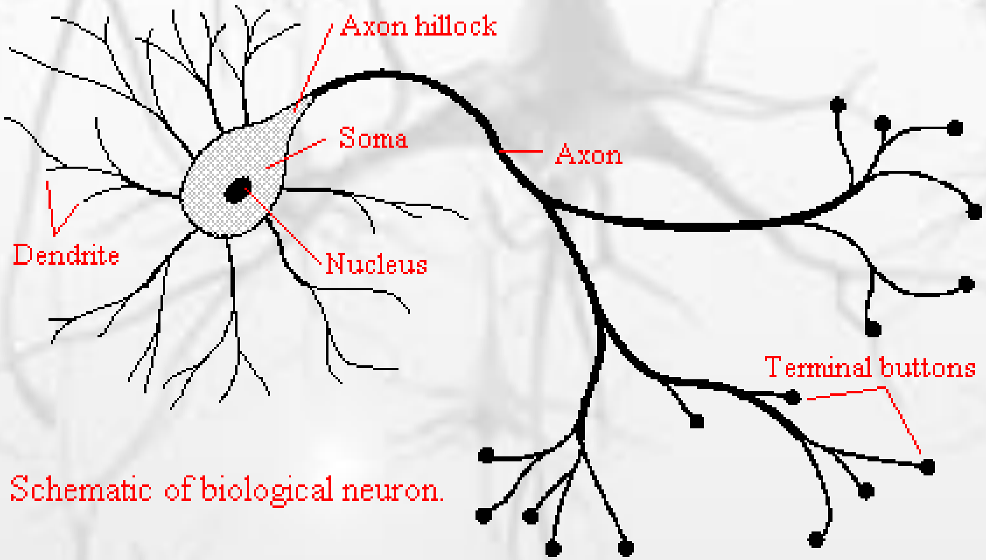
ANN is designed with the goal of building **intelligent machines to solve complex perceptual problems**, such as pattern recognition and optimization, by mimicking the networks of real neurons in the human brain.

Tasks handled by ANNs



Artificial Neural Networks(ANNs) Motivations

❖ANNs are inspired by biological neural networks



A *neuron* (or nerve cell) is a special biological cell, the essence of life, with information processing ability. The introduction of neurons as basic structural constituents of the brain was credited to Ramon y Cajal who won the 1906 Nobel prize for physiology and medicine (shared with Camillo Golgi) for the crucial discovery of the extensive interconnections within the *cerebral cortex*, the portion of the brain where approximately 90% of the neurons in the human are located.

Biological Neuron

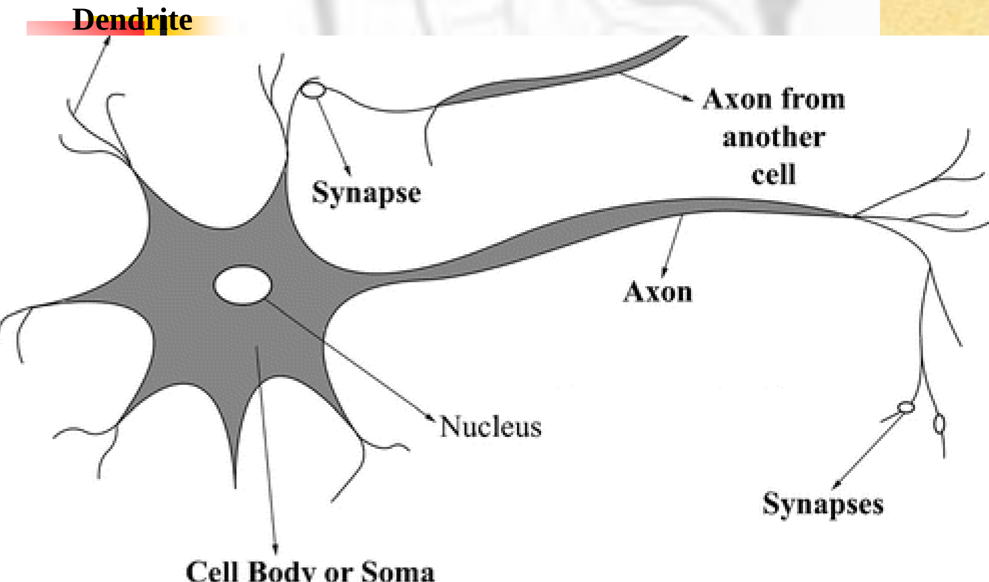
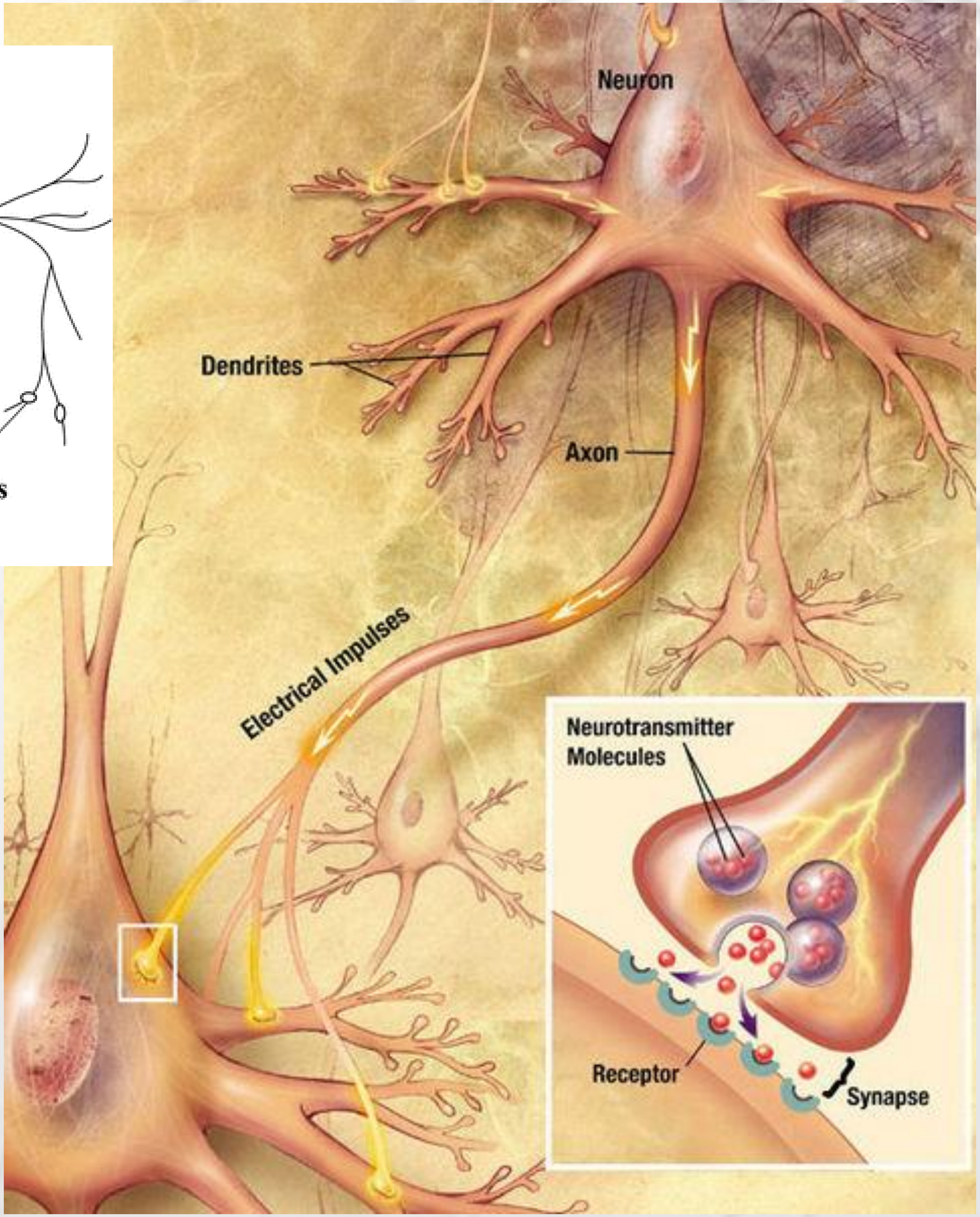
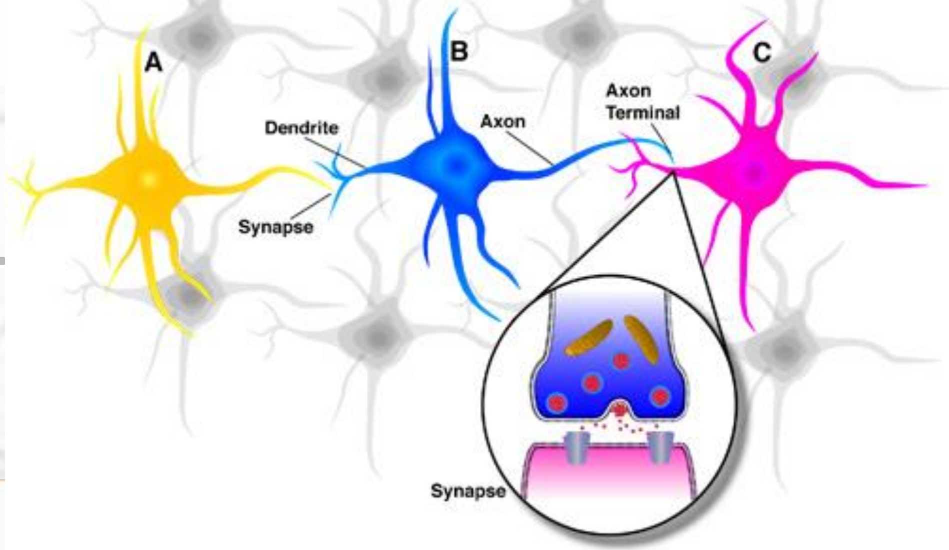
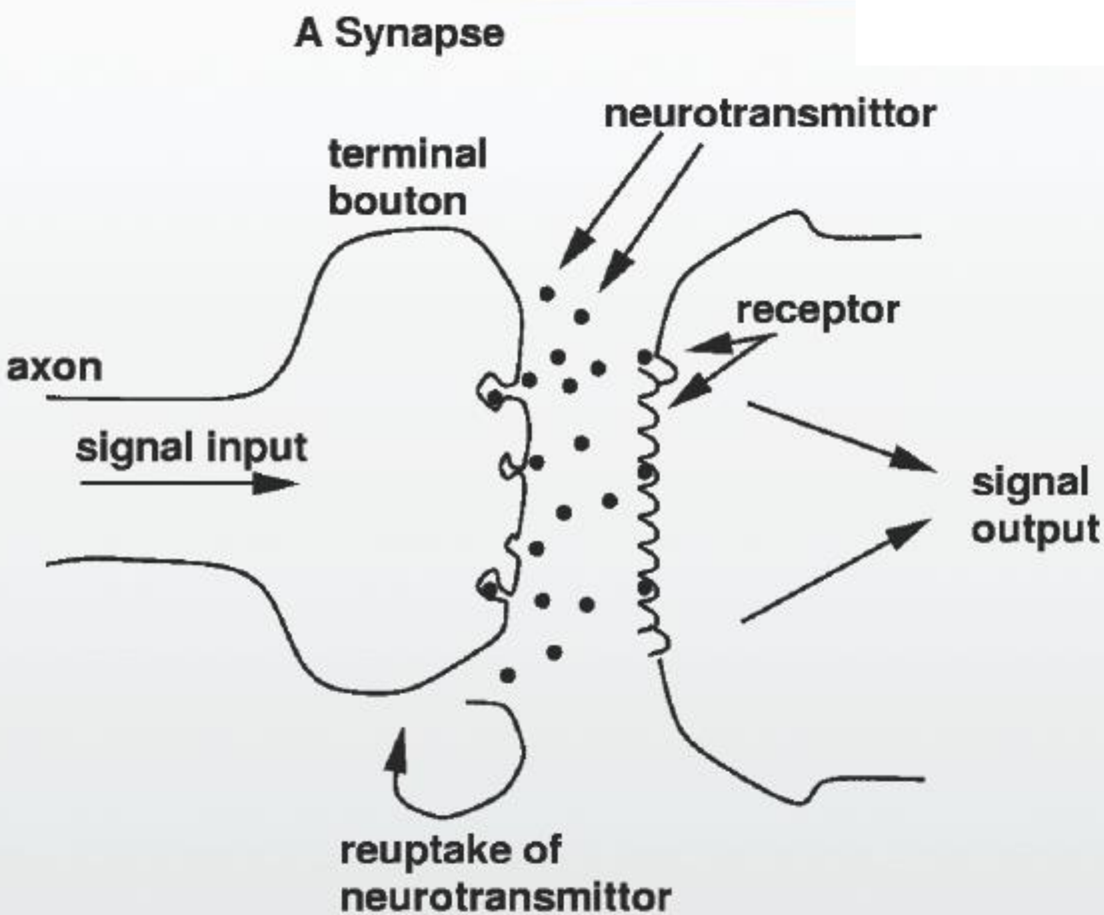


Figure 1.1: A sketch of a biological neuron.



Synapses

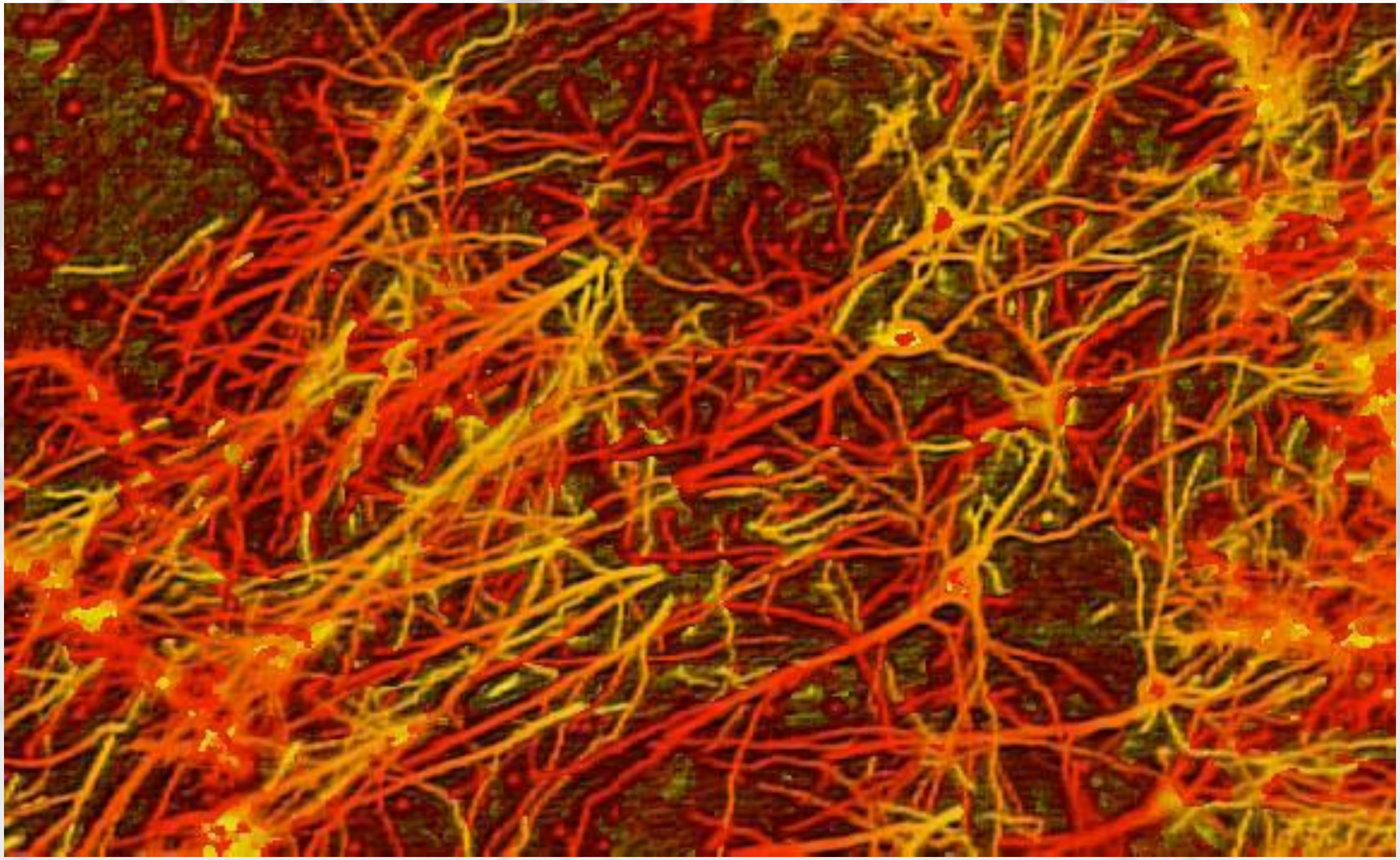




Synapses

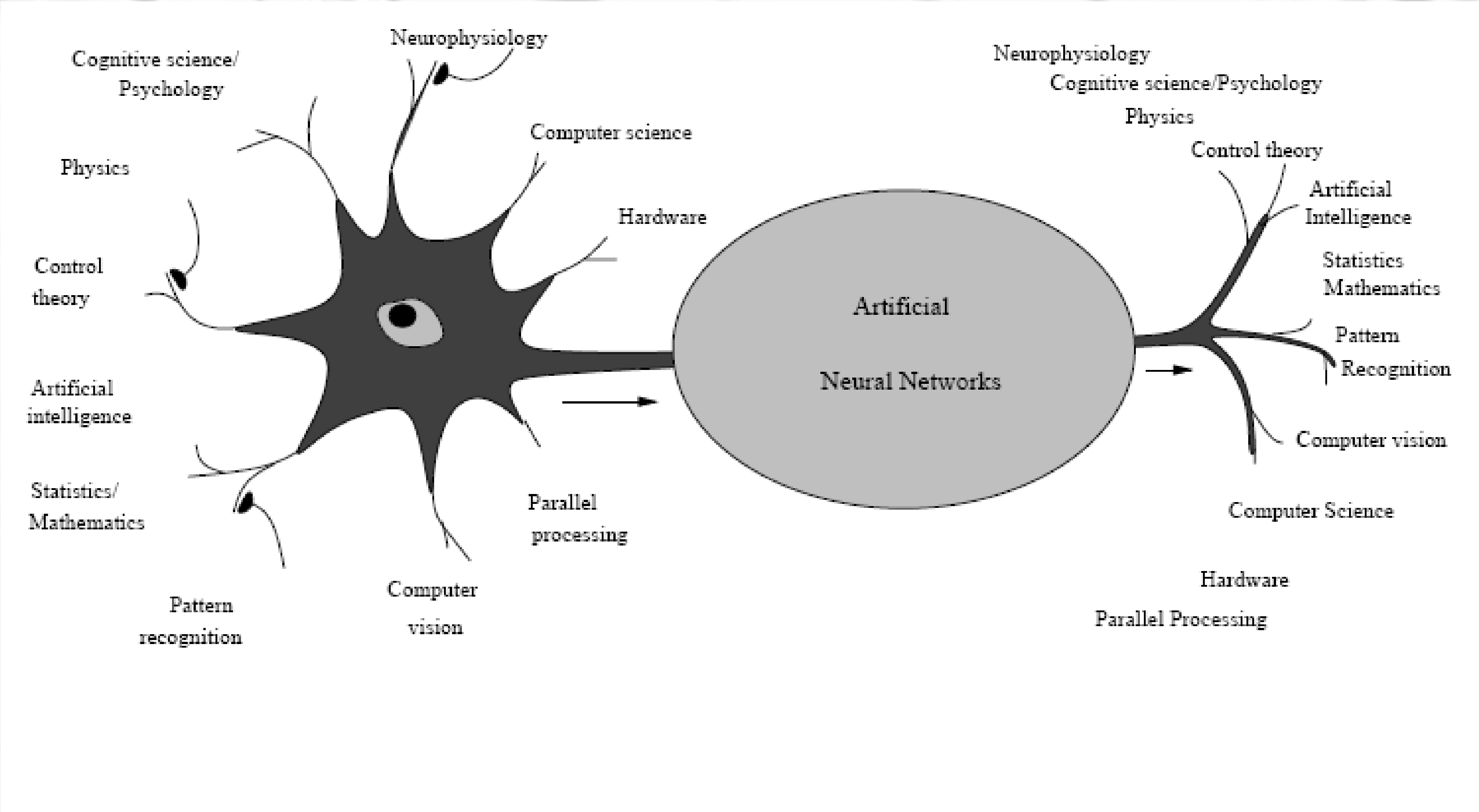
- transmit signal to next neuron
- vary in strength
- *change* strength in response to use (learning!)

A Real Neural Network



Neurons are massively connected, much more complex and denser than today's telephone networks. Each neuron is connected to $10^3 - 10^4$ other neurons. The number of interconnections depends on the location of the neuron in the brain and the type of the neuron. In total, the human brain contains approximately $10^{14} - 10^{15}$ interconnections.

ANN's Relationship with Other Disciplines



Computational Model

$$y = \theta\left(\sum_{j=1}^{n} w_j x_j - u\right),$$

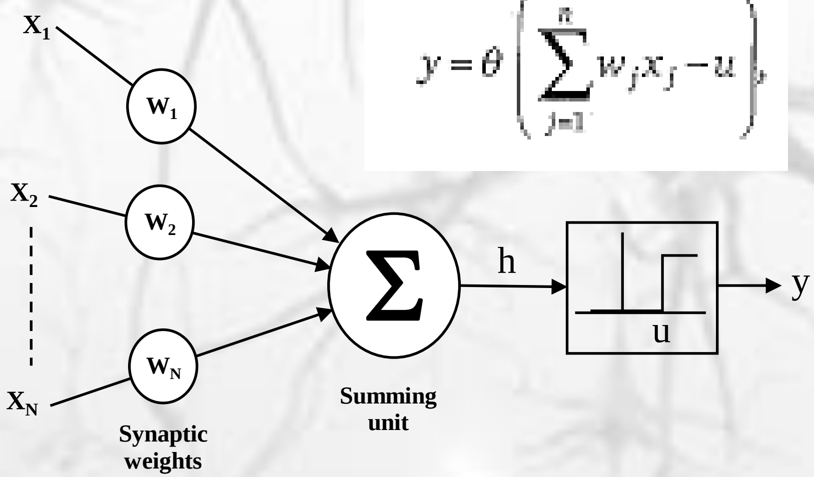


Figure 1.2: McCulloch-Pitts model of a neuron.

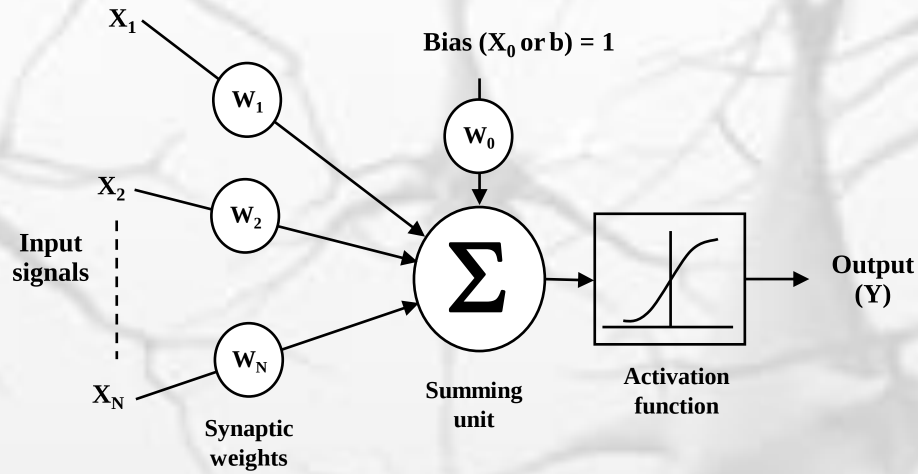


Figure 1.3: Generalized model of an artificial neuron.

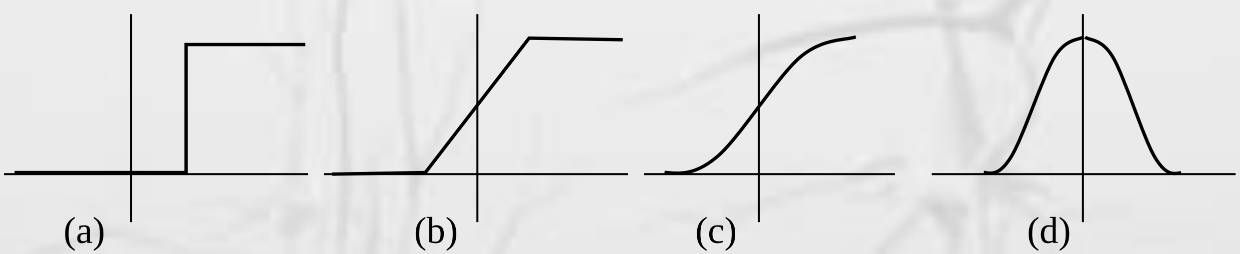


Figure 1.4: Different types of activation functions: (a) threshold, (b) piecewise linear, (c) sigmoid, and (d) Gaussian.

Sigmoid Activation Function

y=1/(1+exp(-ax))

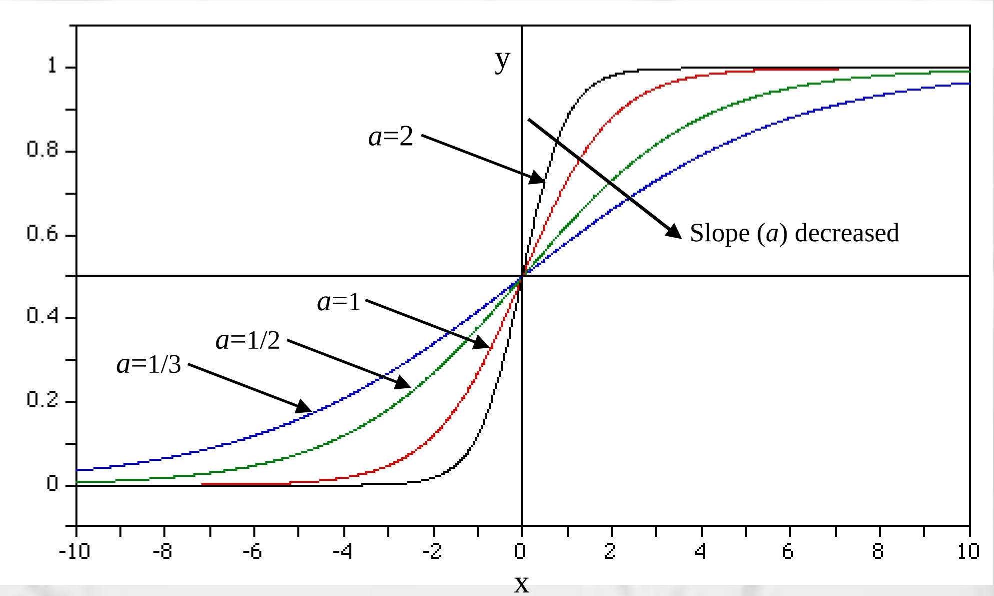
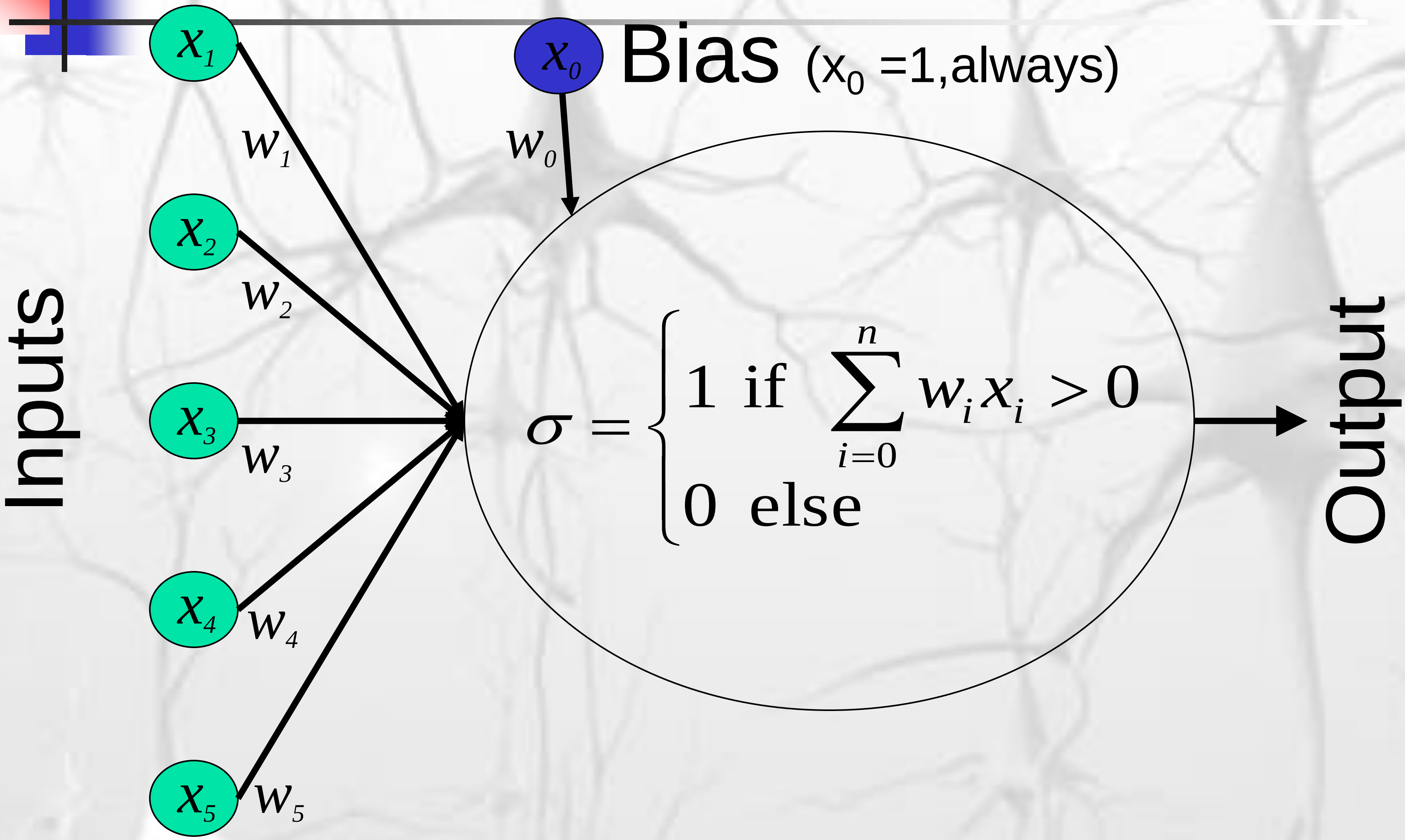
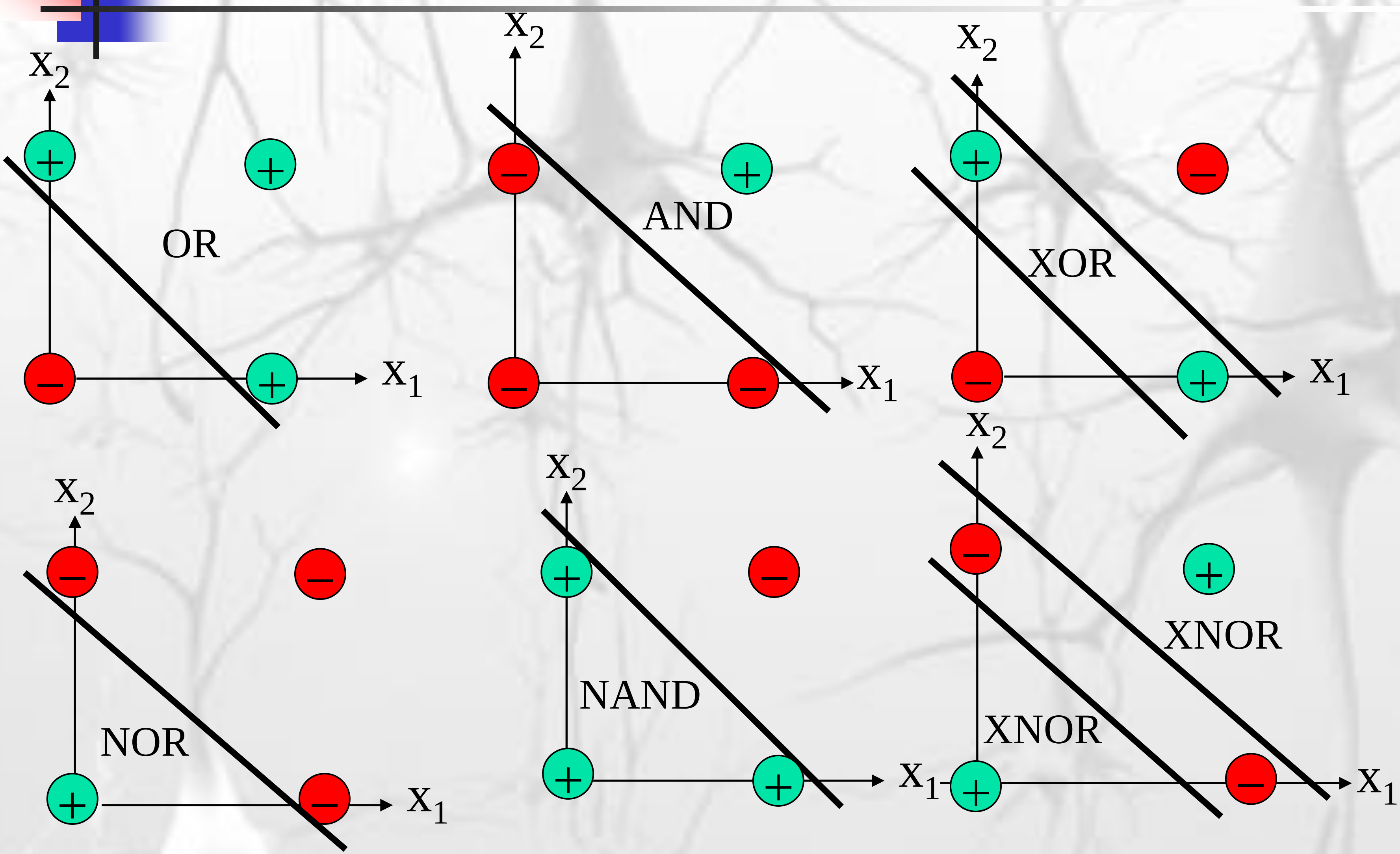


Figure 1.5: Sigmoid function with various slope (a) values.

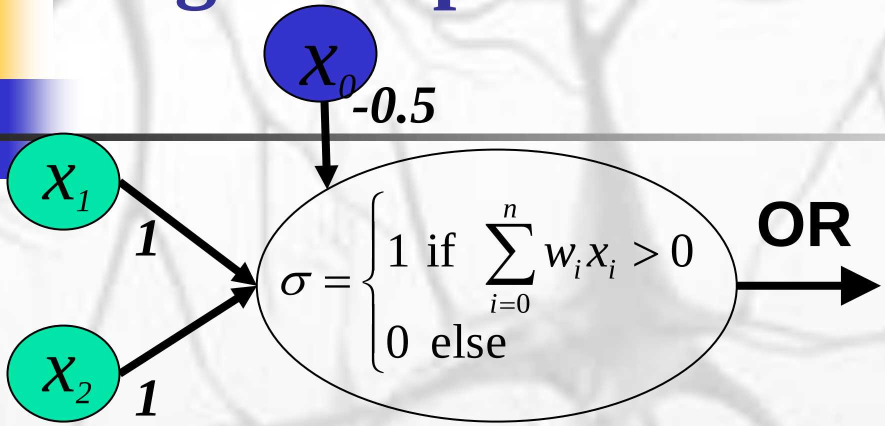
Ability of Single Neuron

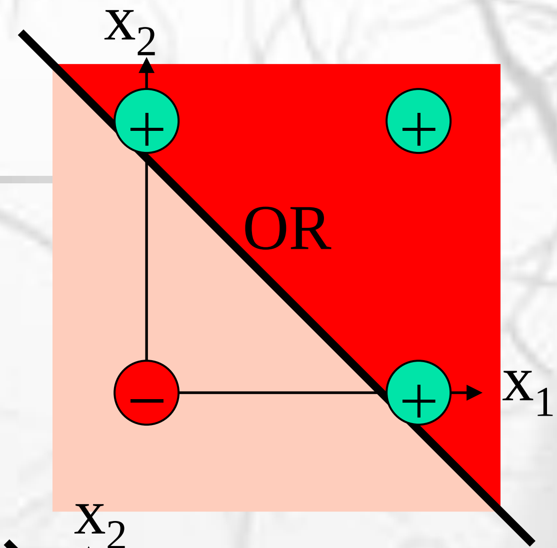


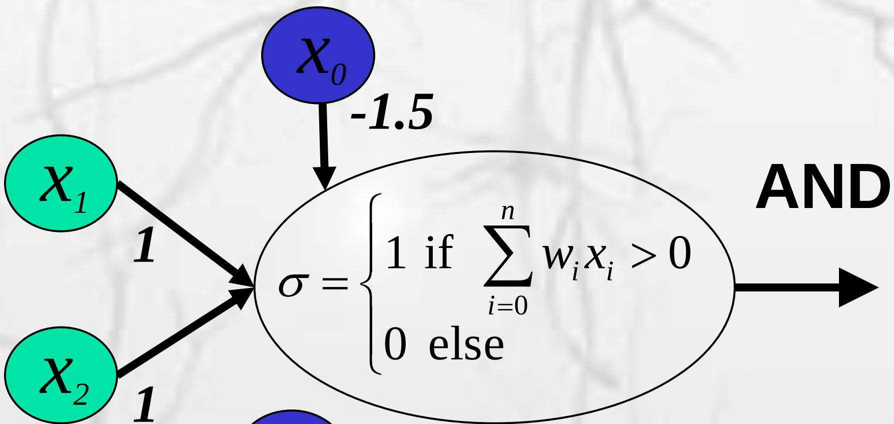
Logical Operators

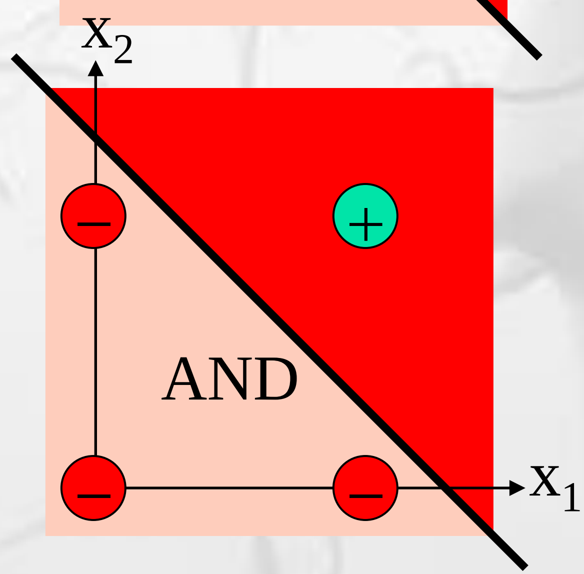


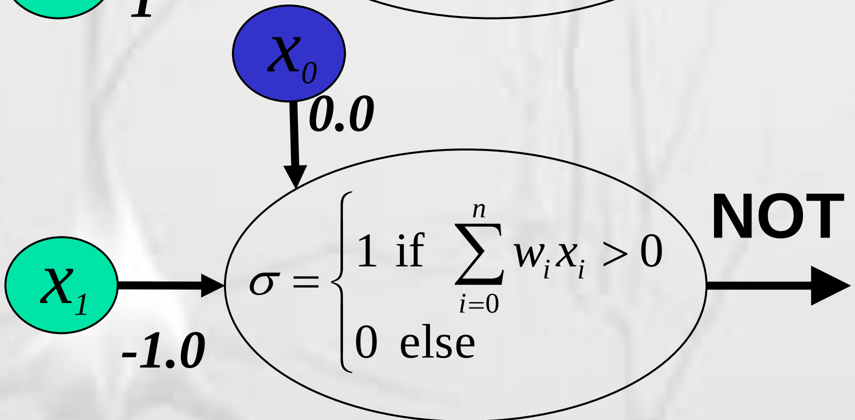
Logical Operators



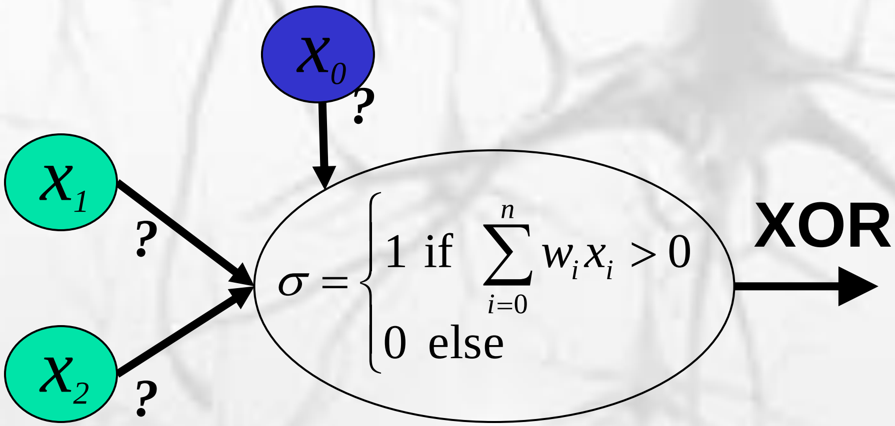


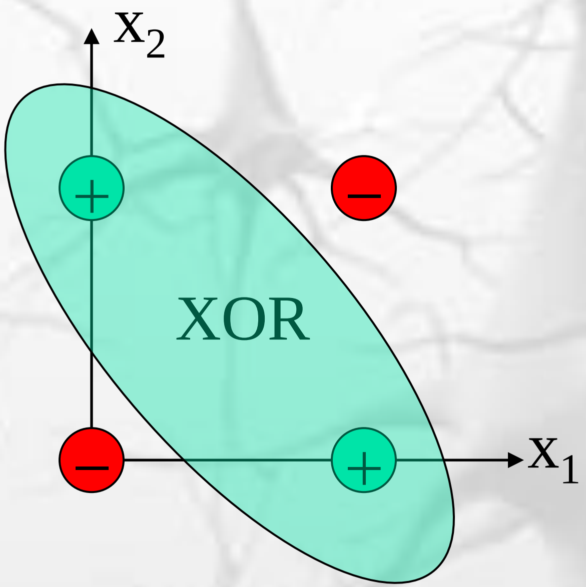






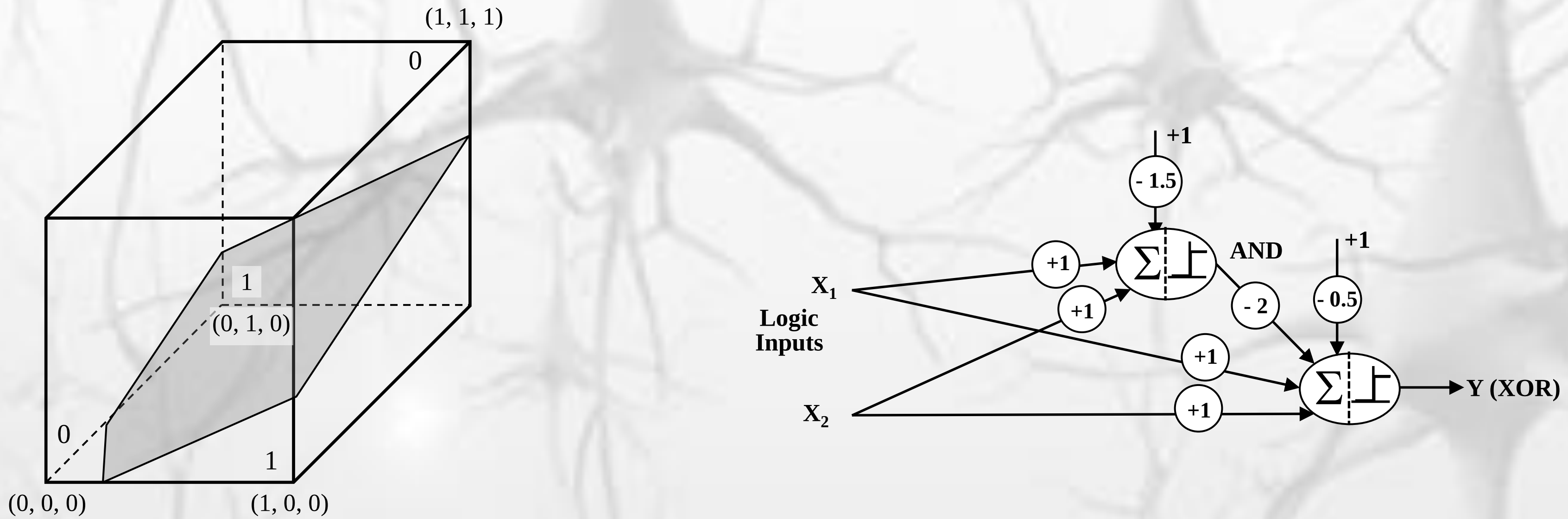
Nonlinear Problem





No arrangement work, not linearly separable. Require two boundary lines.

XOR solution in Higher Dimension



a) Third dimension uplift XOR (1, 1) as (1, 1, 1) in three dimension.

b) XOR solution with additional input from AND gate.

Figure 1.8: Solution of XOR in three dimension.

An additional dimension (input) converts the problem to a linearly separable.

XOR solution with HN

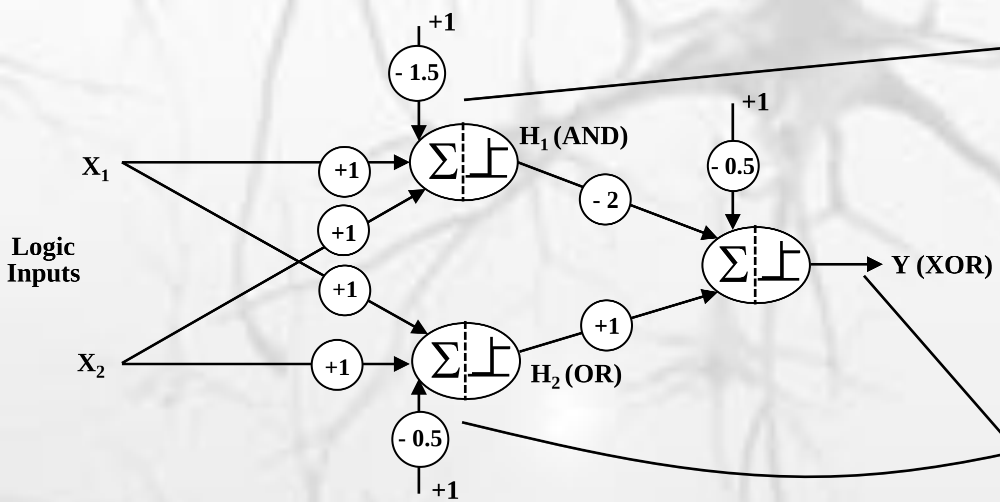
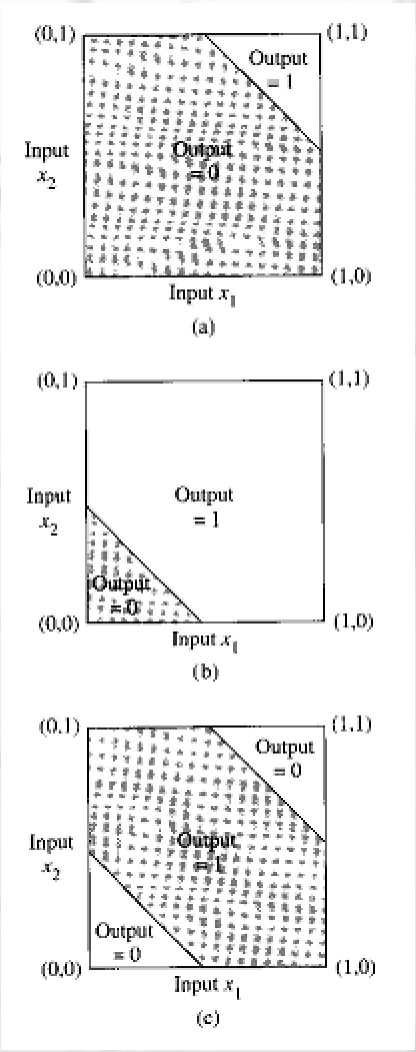


Figure 1.9: XOR logic combining AND and OR logic operations.



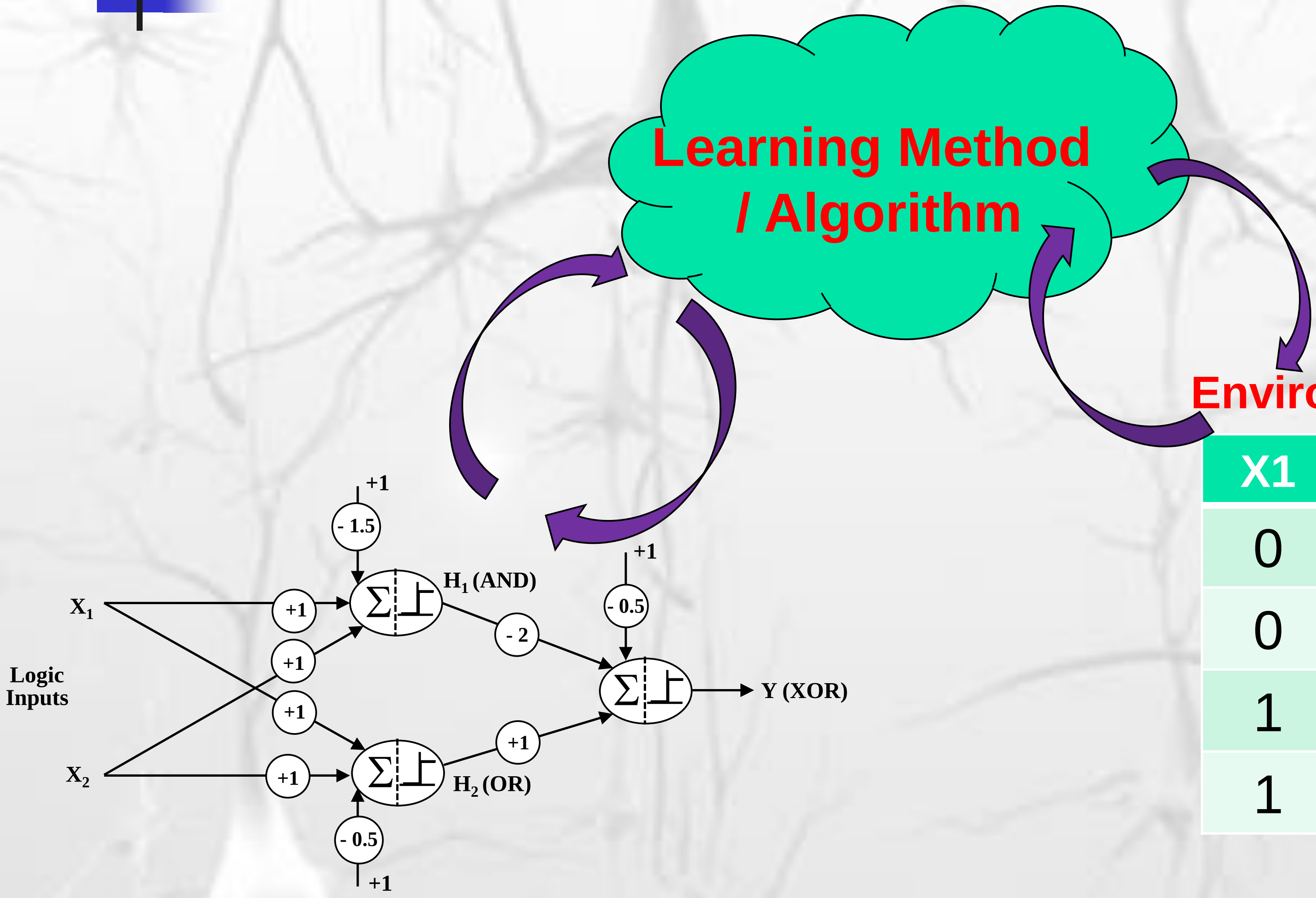
AND

OR

XOR

XOR logic through manipulation by hidden neurons

Learning



Environment / Data

X1	X2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Learning

- The ability to learn is a fundamental trait (characteristic) of intelligence.

- ANN-> Updating NN architecture and connection weights to perform a specific task efficiently.

- ANN ability to automatically learning from examples makes them attractive.

- To understand or design a learning process need: a model of environment or information is available to NN and learning rules.

- A learning algorithm -> procedure for adjusting weights

- Three main learning paradigms-> Supervised (learning with a teacher), unsupervised and hybrid

- Reinforcement learning is variant of supervised learning receives critique on the correctness of network output instead of correct answer.

Learning

- Learning theory must address three fundamental and practical issues:

  1. Capacity – how many patterns can be stored , what functions and decision boundaries a NN can perform

  2. Sample Complexity – Number training patterns needed to train

  3. Computational Complexity – Time required for a learning algorithm to estimate a solution

- Four basic types of Learning rules: Error-Correction, Boltzman, Hebbian, and Competitive.

X1	X2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Network Architecture / Topology

Based on Connection pattern (architecture): feed forward and recurrent (feedback)

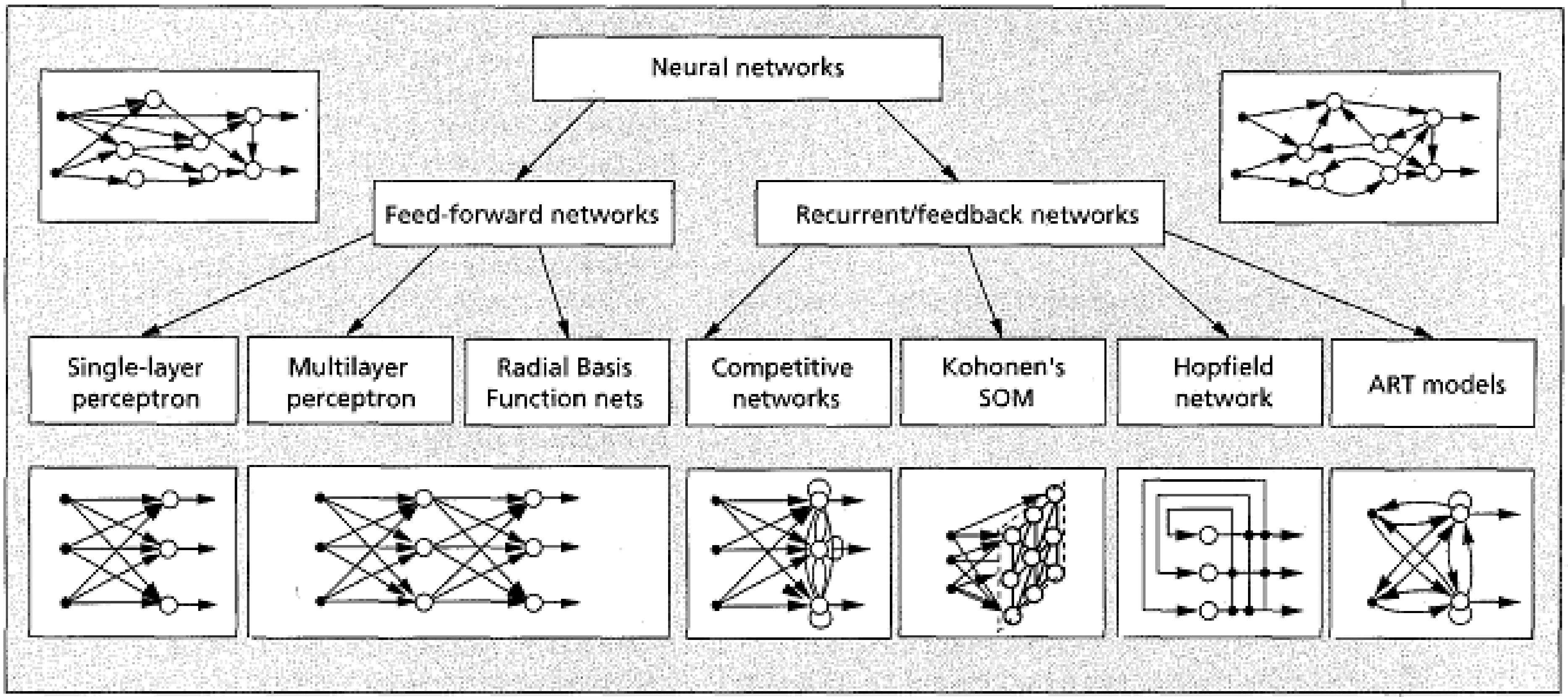


Figure 4. A taxonomy of feed-forward and recurrent/feedback network architectures.

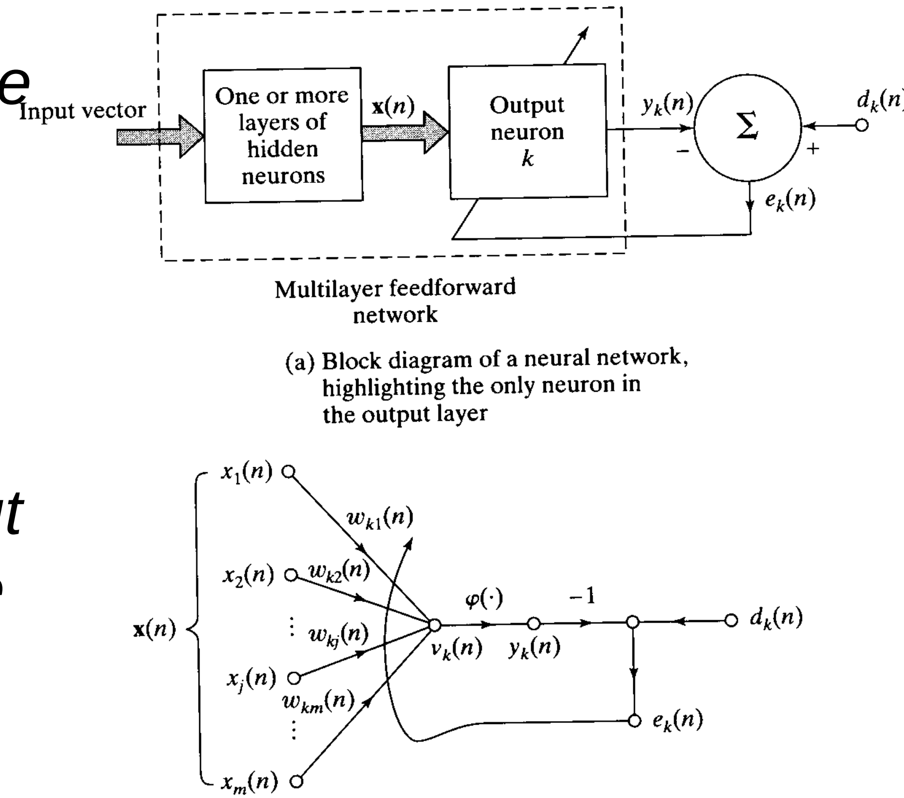
Different architectures require appropriate learning algorithms.

Well Known Learning Algorithms

Paradigm	Learning rule	Architecture	Learning algorithm	Task
Supervised	Error-correction	Single- or multilayer perceptron	Perceptron learning algorithms Back-propagation Adaline and Madaline	Pattern classification Function approximation Prediction, control
	Boltzmann	Recurrent	Boltzmann learning algorithm	Pattern classification
	Hebbian	Multilayer feed-forward	Linear discriminant analysis	Data analysis Pattern classification
	Competitive	Competitive	Learning vector quantization	Within-class categorization Data compression
		ART network	ARTMap	Pattern classification Within-class categorization
Unsupervised	Error-correction	Multilayer feed-forward	Sammon's projection	Data analysis
	Hebbian	Feed-forward or competitive	Principal component analysis	Data analysis Data compression
		Hopfield Network	Associative memory learning	Associative memory
	Competitive	Competitive	Vector quantization	Categorization Data compression
		Kohonen's SOM	Kohonen's SOM	Categorization Data analysis
		ART networks	ART1, ART2	Categorization
Hybrid	Error-correction and competitive	RBF network	RBF learning algorithm	Pattern classification Function approximation Prediction, control

Learning Rule: Error-Correction

- *error signal = desired response – output signal*
- $e_k(n) = d_k(n) - y_k(n)$
- $e_k(n)$ *actuates a control mechanism to make the output signal* $y_k(n)$ *come closer to the desired response* $d_k(n)$ *in step by step manner*



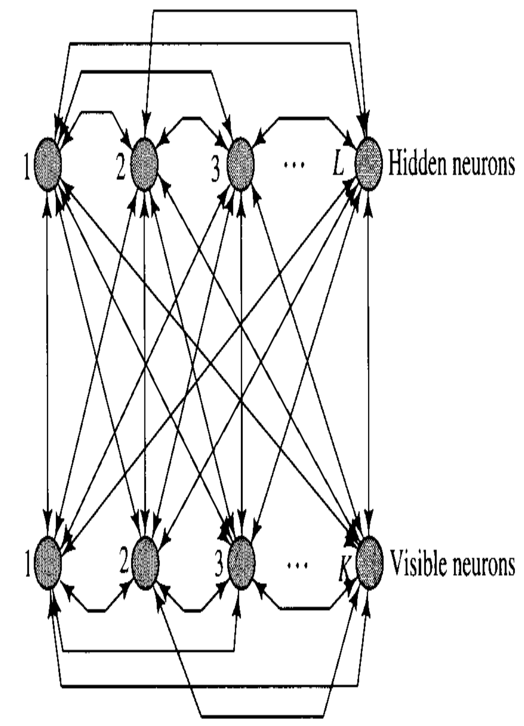
(a) Block diagram of a neural network, highlighting the only neuron in the output layer

(b) Signal-flow graph of output neuron

FIGURE 2.1 Illustrating error-correction learning.

Learning Rule: Boltzmann

- The primary goal of Boltzmann learning is to produce a neural network that correctly models input patterns according to a Boltzmann distribution.
- The Boltzmann machine consists of stochastic neurons. A stochastic neuron resides in one of two possible states (± 1) in a probabilistic manner.
- The use of symmetric synaptic connections between neurons.
- The stochastic neurons partition into two functional groups: visible and hidden.
- During the training phase of the network , the visible neurons are all clamped onto specific states determined by the environment.
- The hidden neurons always operate freely; they are used to explain underlying constraints contained in the environmental input vectors.



Learning Rule: Hebbian

1. *If two neurons on either side of synapse (connection) are activated simultaneously, then the strength of that synapse is selectively increased.*

2. *If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.*

A Hebbian synapse increases its strength with positively correlated presynaptic and postsynaptic signals, and decreases its strength when signals are either uncorrelated or negatively correlated.

Learning Rule: Competitive

- *The output neurons of a neural network compete among themselves to become active.*
- *- a set of neurons that are all the same (excepts for synaptic weights)*
- *- a limit imposed on the strength of each neuron*
- *- a mechanism that permits the neurons to compete -> a winner-takes-all*

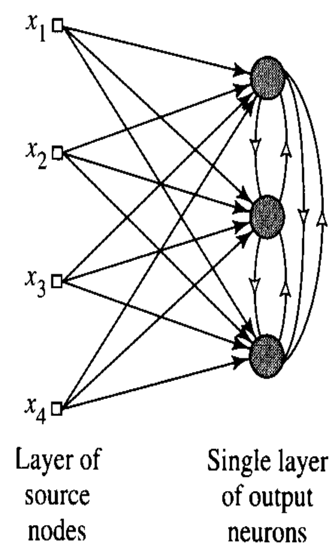


FIGURE 2.4 Architectural graph of a simple competitive learning network with feedforward (excitatory) connections from the source nodes to the neurons, and lateral (inhibitory) connections among the neurons; the lateral connections are signified by open arrows.

- *The standard competitive learning rule*
- $\Delta w_{kj} = \eta(x_j - w_{kj})$ *if neuron k wins the competition*

  $= 0$ *if neuron k loses the competition*

Learning Paradigms : Learning with a Teacher

Learning with a Teacher (=supervised learning)

The teacher has knowledge of the environment

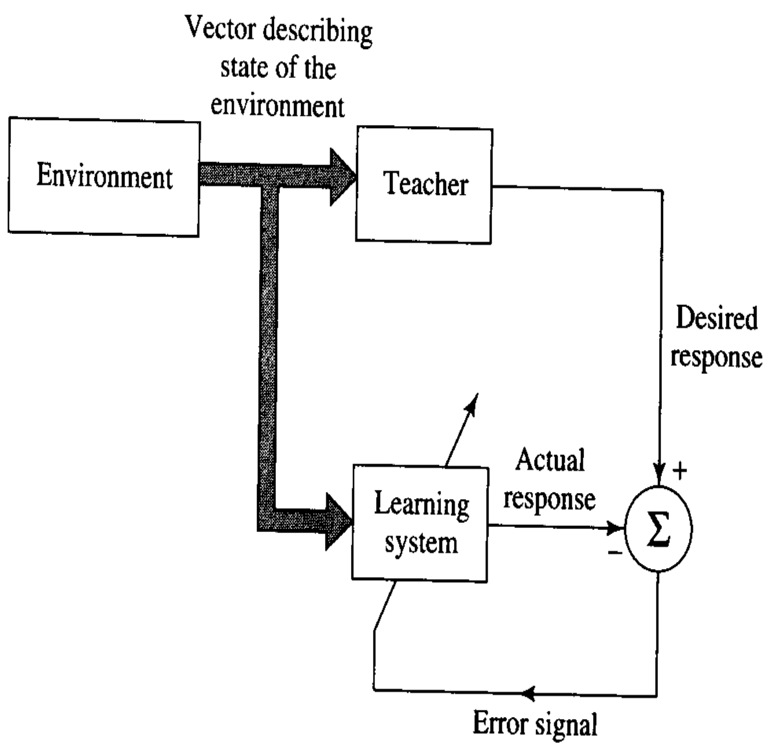


FIGURE 2.6 Block diagram of learning with a teacher.

Learning Paradigms: Learning without a Teacher

Learning without a Teacher: no labeled examples available of the function to be learned.

1) Reinforcement learning

2) Unsupervised learning

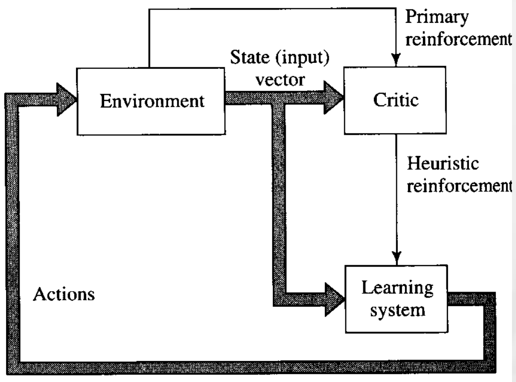


FIGURE 2.7 Block diagram of reinforcement learning.

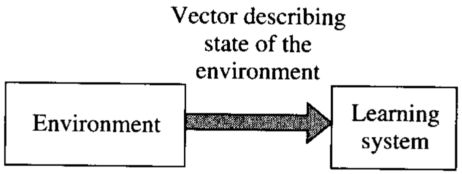


FIGURE 2.8 Block diagram of unsupervised learning.

Learning Paradigms: Reinforcement

Reinforcement learning:

The learning of input-output mapping is performed through continued interaction with the environment in oder to minimize a scalar index of performance.

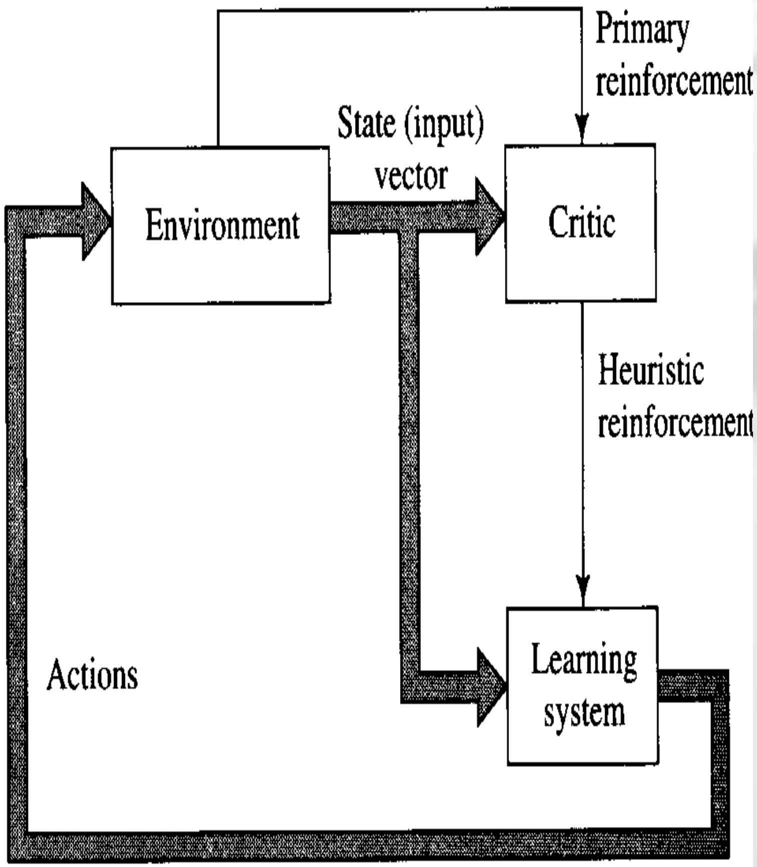


FIGURE 2.7 Block diagram of reinforcement learning.

Learning Paradigms: Unsupervised

- *Unsupervised Learning: There is no external teacher or critic to oversee the learning process.*

- *The provision is made for a task independent measure of the quality of representation that the network is required to learn.*

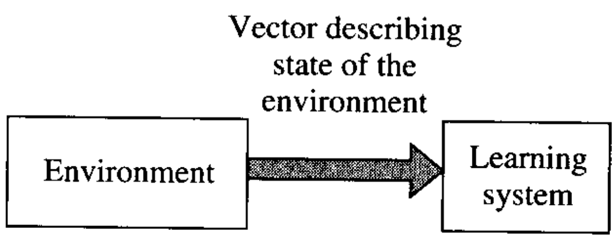


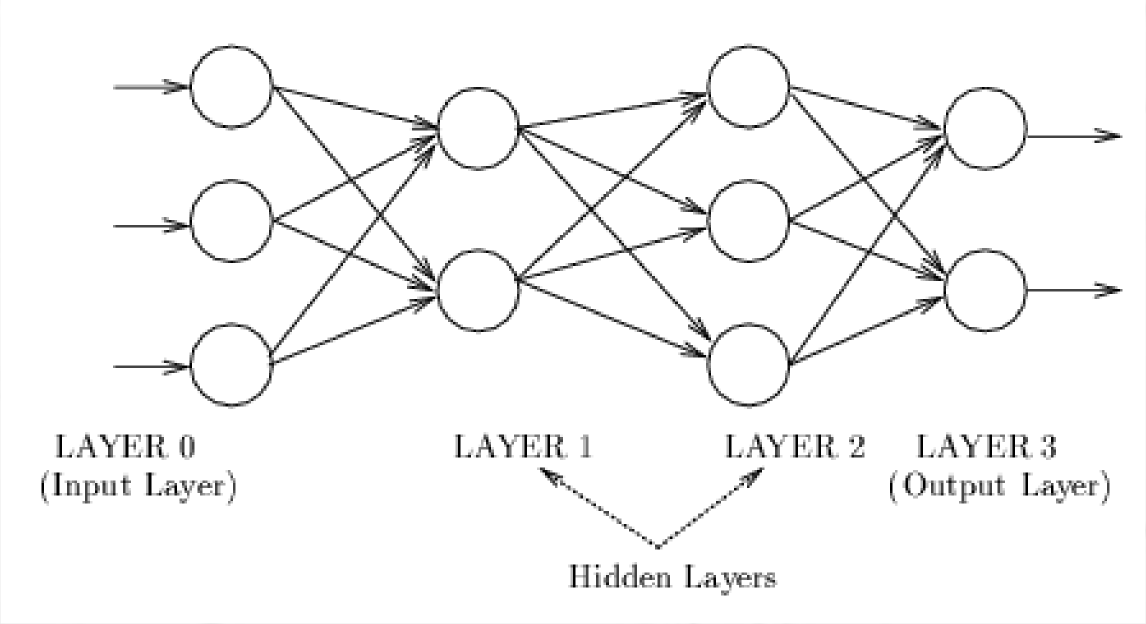
FIGURE 2.8 Block diagram of unsupervised learning.

Training of Multilayer Feed-Forward Neural Networks : Back-propagtaion

Multilayer Feed-Forward Neural Networks

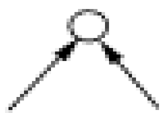
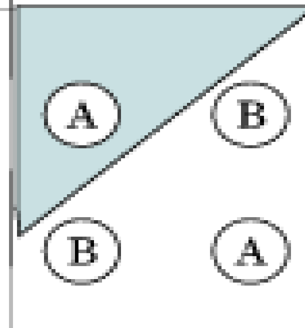
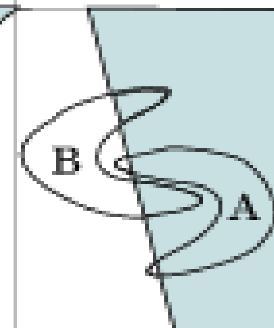
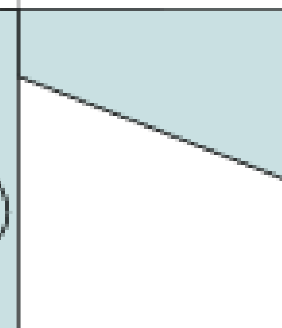
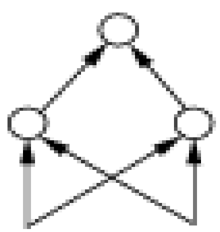
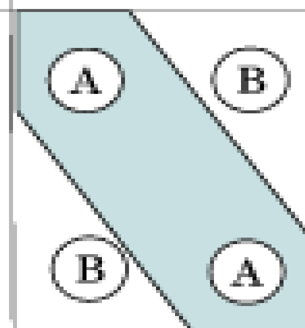
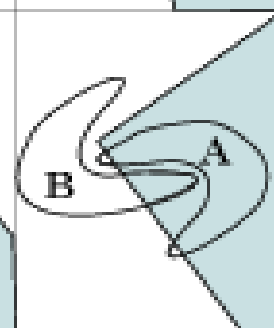
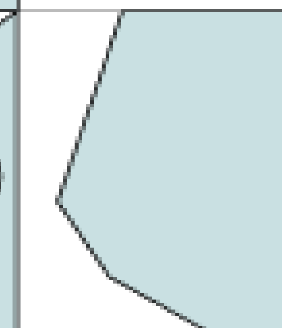
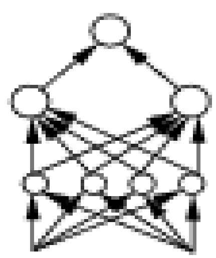
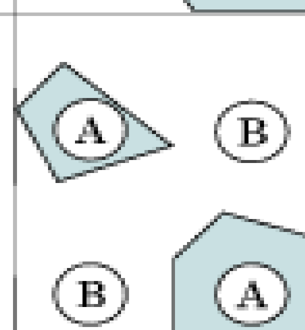
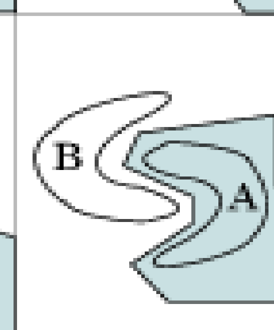
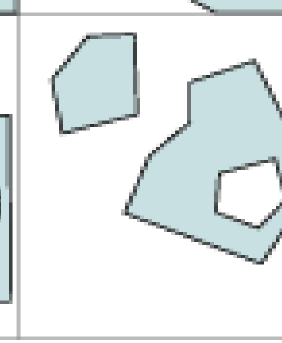
Feed -forward Networks

- ***A connection is allowed from a node in layer* i *only to nodes in layer* i + *1.***
- ***Most widely used architecture.***

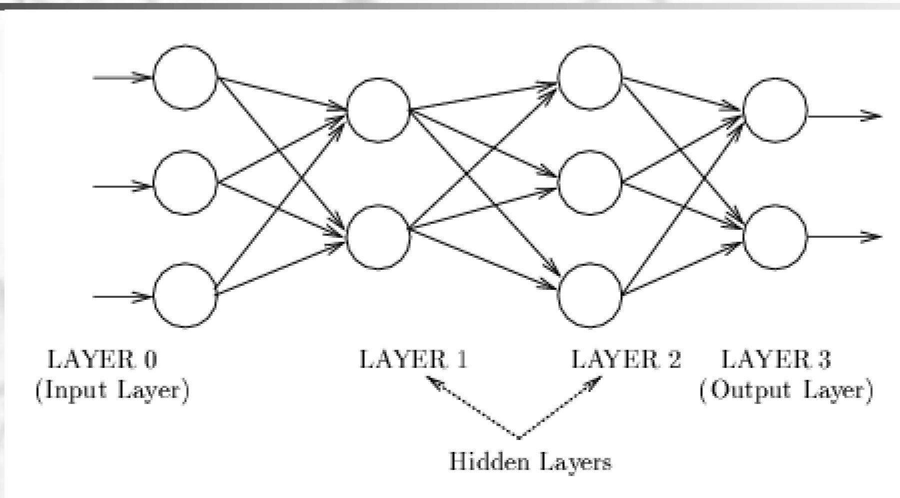


Conceptually, nodes at higher levels successively abstract features from preceding layers

Geometric interpolation of the role of HN

Structure	Types of decision regions	Exclusive OR problem	Classes with meshed regions	Most general region shapes
Single-layer 	Half Plane (Bounded by hyperplane)			
Two-layer 	Convex (Open or closed regions)			
Three-layer 	Arbitrary (Complexity limited by number of neurons)			

How to get appropriate weight set?



Back-propagation is the famous algorithm for training feed-forward networks

http://en.wikipedia.org/wiki/Backpropagation

Backpropagation, or propagation of error, is a common method of teaching artificial neural networks how to perform a given task.
It was first described by Paul Werbos in 1974, but it wasn't until 1986, through the work of David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams, that it gained recognition, and it led to a **"renaissance"** in the field of artificial neural network research.

letters to nature

Nature **323**, 533 - 536 (09 October 1986); doi:10.1038/323533a0

Learning representations by back-propagating errors

DAVID E. RUMELHART[*], GEOFFREY E. HINTON[†] & RONALD J. WILLIAMS[*]

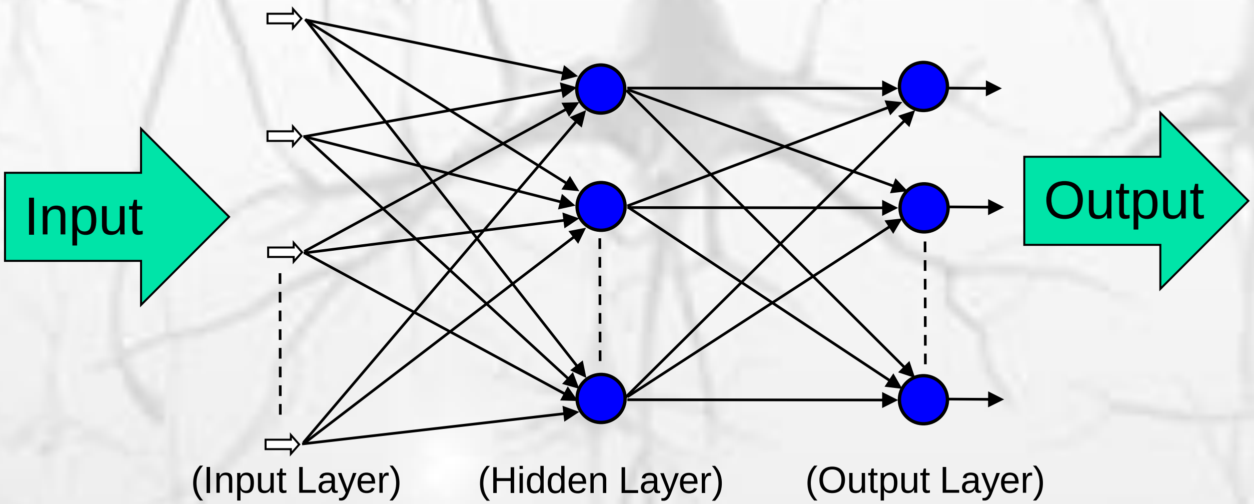
[*]Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
[†]Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA
[†]To whom correspondence should be addressed.

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

References

1. Rosenblatt, F. *Principles of Neurodynamics* (Spartan, Washington, DC, 1961).
2. Minsky, M. L. & Papert, S. *Perceptrons* (MIT, Cambridge, 1969).
3. Le Cun, Y. *Proc. Cognitiva* **85**, 599–604 (1985).

Back-propagation(BP)

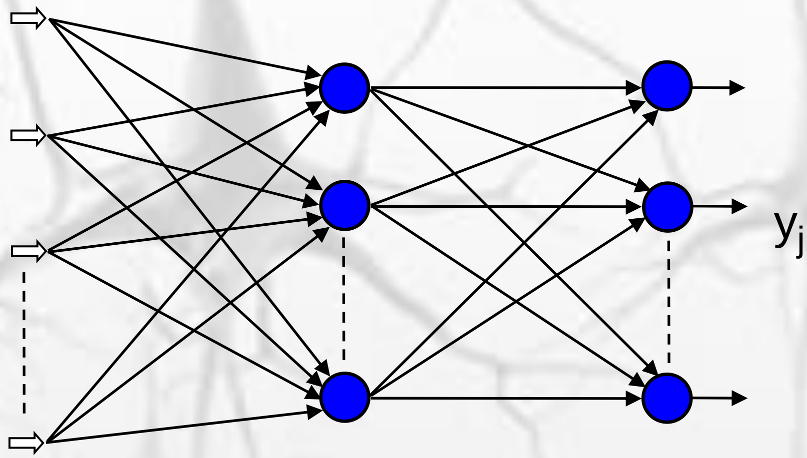


Error = Desired Output - Actual Output

Forward Pass: Error is calculated based on desired and actual output.

Backward Pass: Synaptic weights are adjusted based on calculated error.

Back-propagation



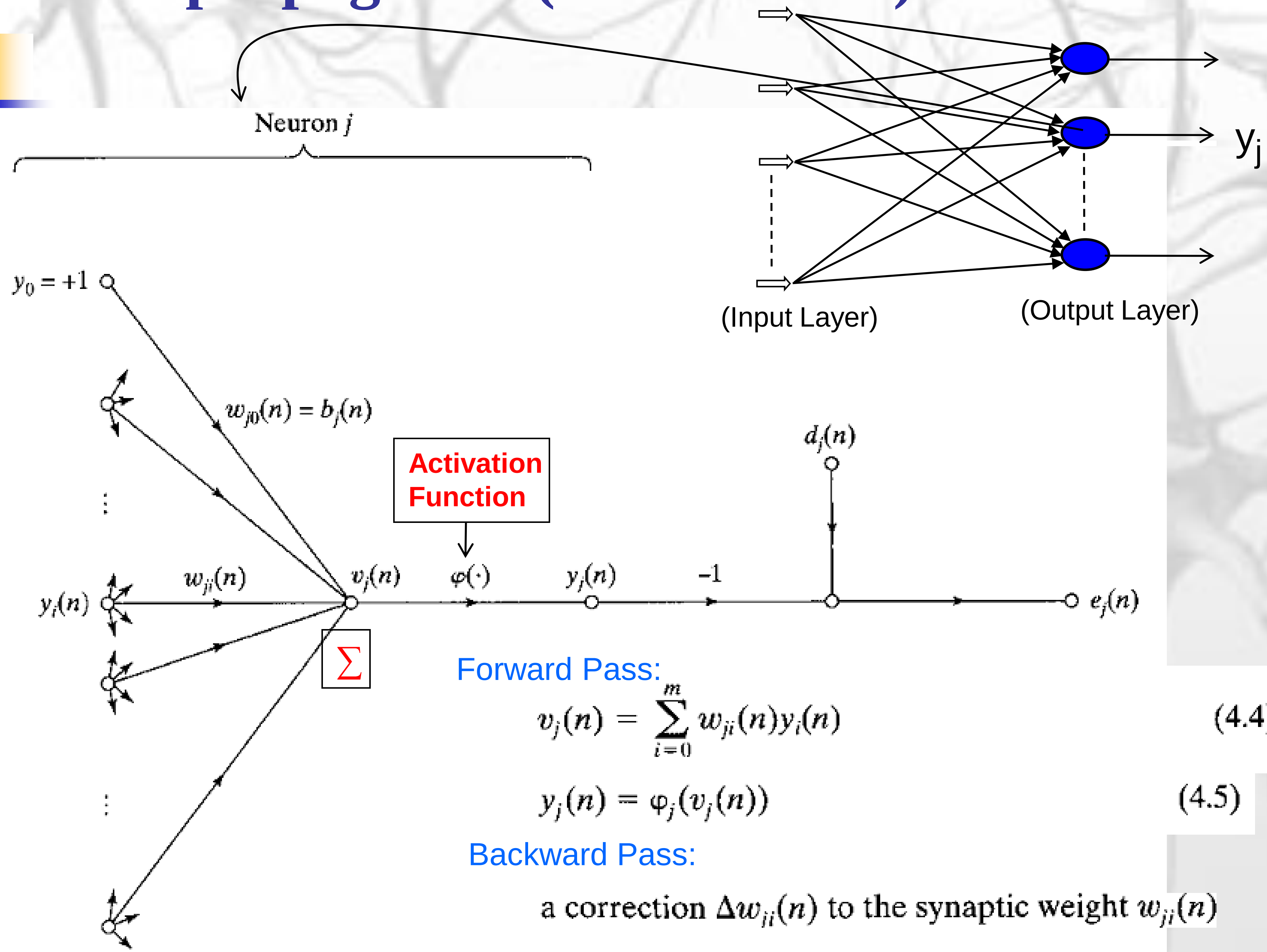
$$e_j(n) = d_j(n) - y_j(n), \qquad \text{neuron } j \text{ is an output node} \qquad (4.1)$$

$$\mathscr{E}(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n) \qquad (4.2)$$

$$\mathscr{E}_{\text{av}} = \frac{1}{N} \sum_{n=1}^{N} \mathscr{E}(n) \qquad (4.3)$$

performance. The objective of the learning process is to adjust the free parameters of the network to minimize $\mathscr{E}_{\text{av}}$. ~~To do this minimization, we use an approximation similar~~

Back-propagation (Without HN)



Forward Pass:

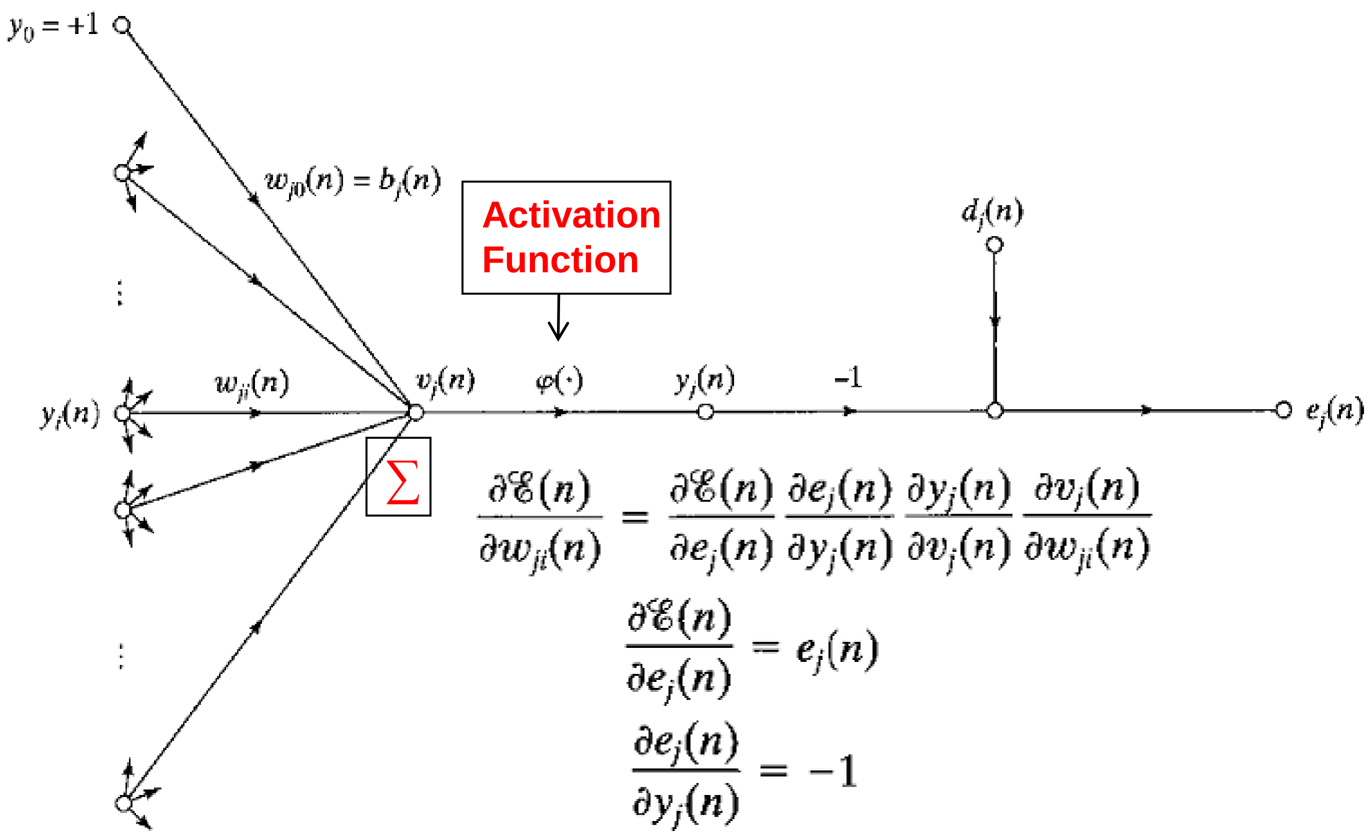
$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \qquad (4.4)$$

$$y_j(n) = \varphi_j(v_j(n)) \qquad (4.5)$$

Backward Pass:

a correction $\Delta w_{ji}(n)$ to the synaptic weight $w_{ji}(n)$

FIGURE 4.3 Signal-flow graph highlighting the details of output neuron j.



$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathscr{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{4.6}$$

$$\frac{\partial \mathscr{E}(n)}{\partial e_j(n)} = e_j(n) \tag{4.7}$$

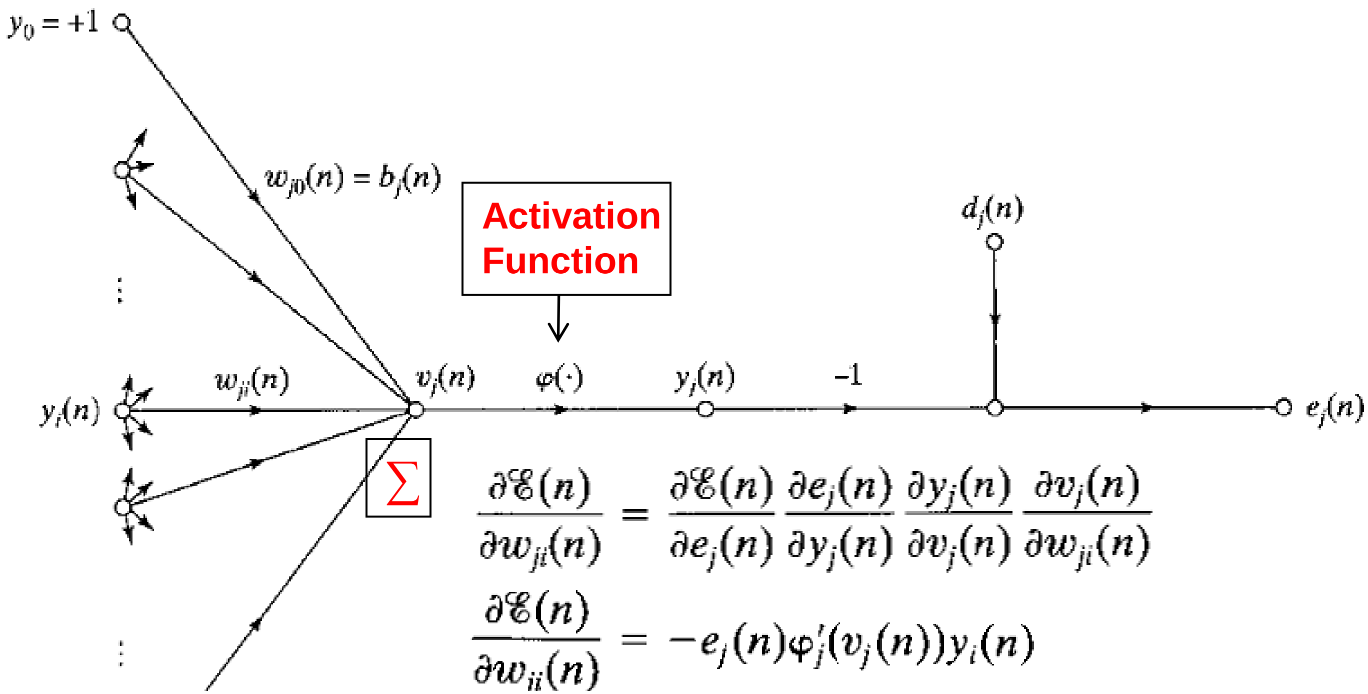
$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{4.8}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) \tag{4.9}$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{4.10}$$

$$\boxed{\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n) \qquad (4.11)}$$

Depends on Activation Function



$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathscr{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \qquad (4.6)$$

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n) \qquad (4.11)$$

The correction $\Delta w_{ji}(n)$ applied to $w_{ji}(n)$ is defined by the *delta rule*:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} \qquad (4.12)$$

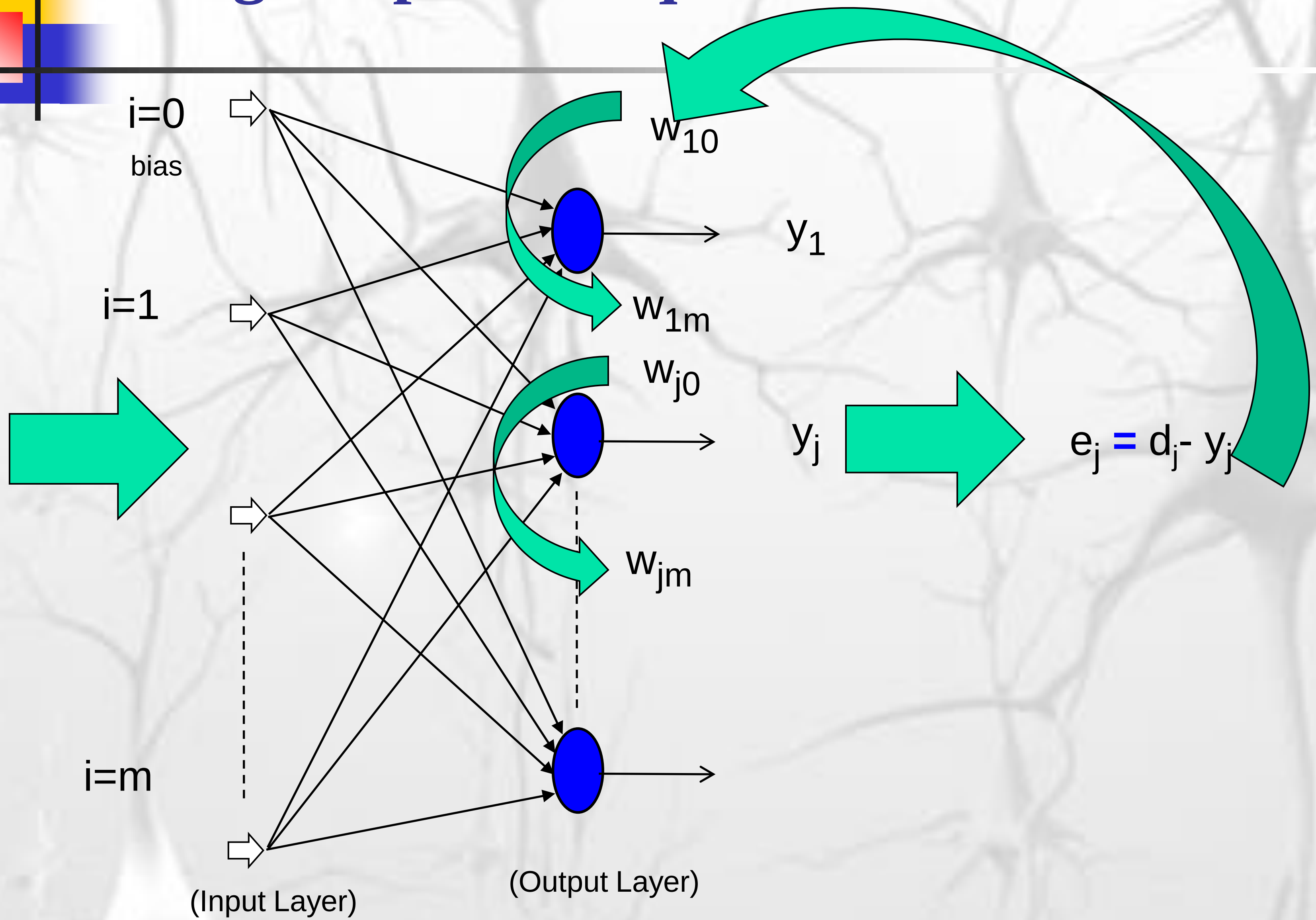
where η is the *learning-rate parameter* of the back-propagation algorithm. The use of the minus sign in Eq. (4.12) accounts for *gradient descent* in weight space (i.e., seeking a direction for weight change that reduces the value of $\mathscr{E}(n)$). Accordingly, the use of Eq. (4.11) in (4.12) yields

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \qquad (4.13)$$

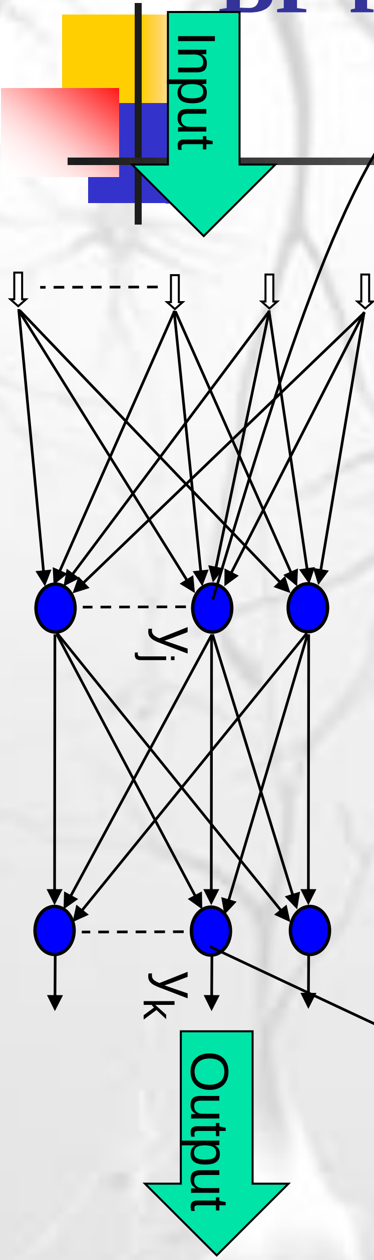
where the *local gradient* $\delta_j(n)$ is defined by

$$\delta_j(n) = -\frac{\partial \mathscr{E}(n)}{\partial v_j(n)} = -\frac{\partial \mathscr{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\varphi_j'(v_j(n)) \qquad (4.14)$$

Weight Update Sequence in BP



BP for Multilayer (With HN)



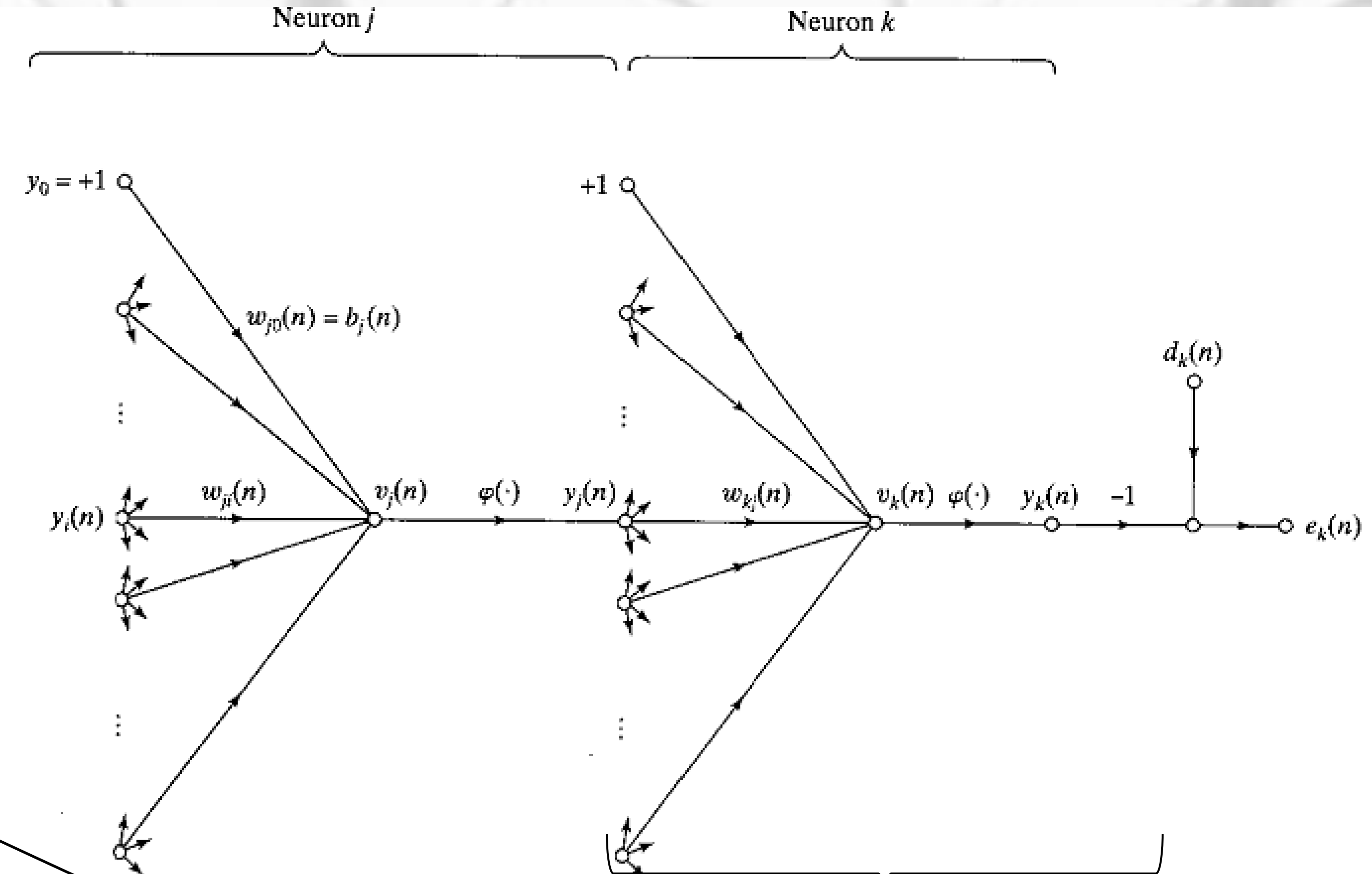
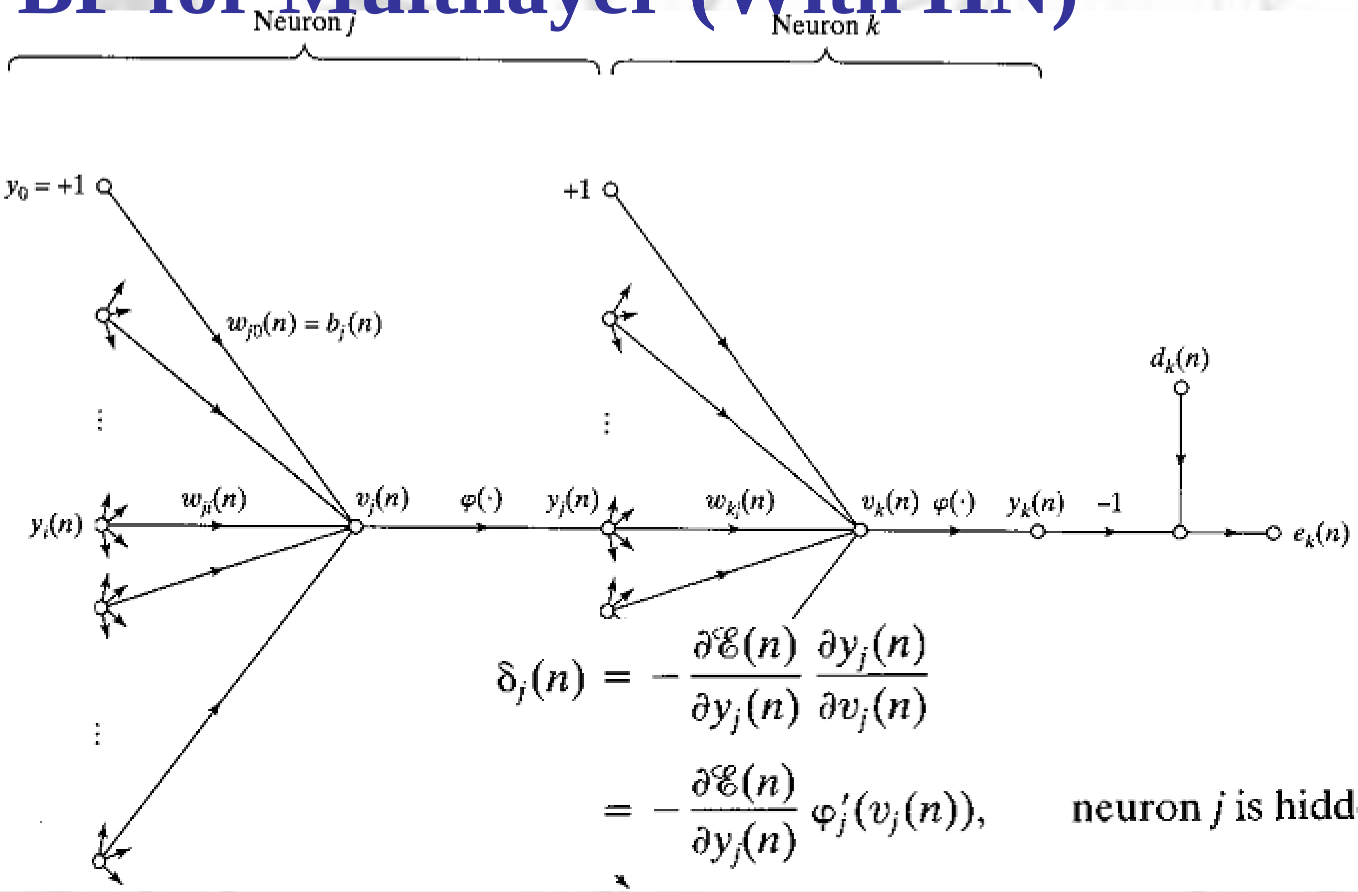


FIGURE 4.4 Signal-flow graph highlighting the details of output neuron *k* connected to hidden neuron *j*.

BP for Multilayer (With HN)



$$\begin{aligned}\delta_j(n) &= -\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \\ &= -\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} \varphi_j'(v_j(n)), \qquad \text{neuron } j \text{ is hidden}\end{aligned} \tag{4.15}$$

$$\mathscr{E}(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n), \qquad \text{neuron } k \text{ is an output node} \tag{4.16}$$

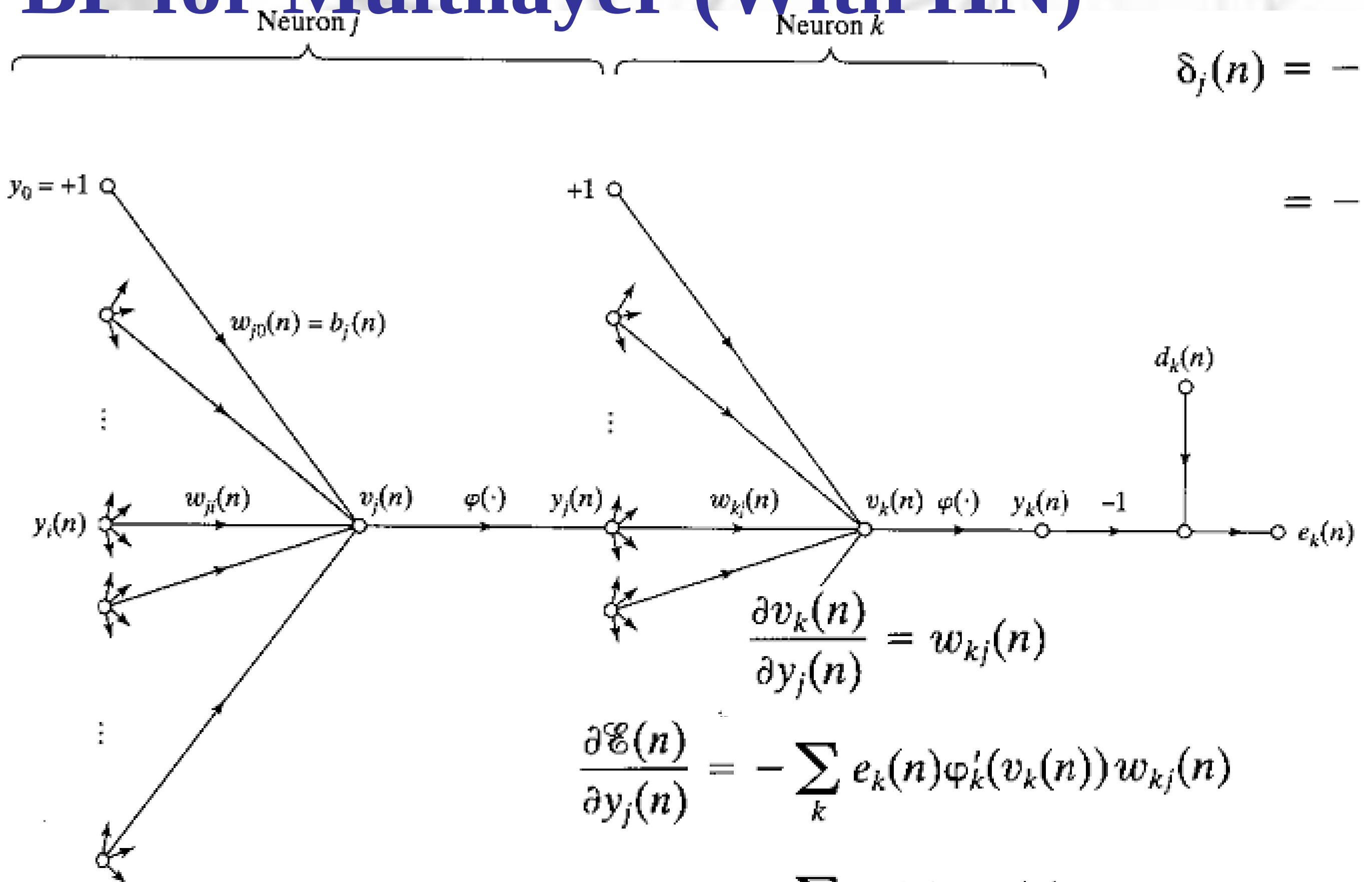
$$\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \tag{4.17}$$

$$\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{4.18}$$

$$\begin{aligned}e_k(n) &= d_k(n) - y_k(n) \\ &= d_k(n) - \varphi_k(v_k(n)),\end{aligned} \tag{4.19}$$

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n) \tag{4.21}$$

BP for Multilayer (With HN)



$$\delta_j(n) = -\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} \varphi_j'(v_j(n)),$$

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{4.22}$$

$$\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} = -\sum_k e_k(n)\varphi_k'(v_k(n))w_{kj}(n) = -\sum_k \delta_k(n)w_{kj}(n) \tag{4.23}$$

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n), \qquad \text{neuron } j \text{ is hidden} \tag{4.24}$$

$$\begin{pmatrix} \text{Weight} \\ \text{correction} \\ \Delta w_{ji}(n) \end{pmatrix} = \begin{pmatrix} \text{learning-} \\ \text{rate parameter} \\ \eta \end{pmatrix} \cdot \begin{pmatrix} \text{local} \\ \text{gradient} \\ \delta_j(n) \end{pmatrix} \cdot \begin{pmatrix} \text{input signal} \\ \text{of neuron } j \\ y_i(n) \end{pmatrix} \tag{4.25}$$

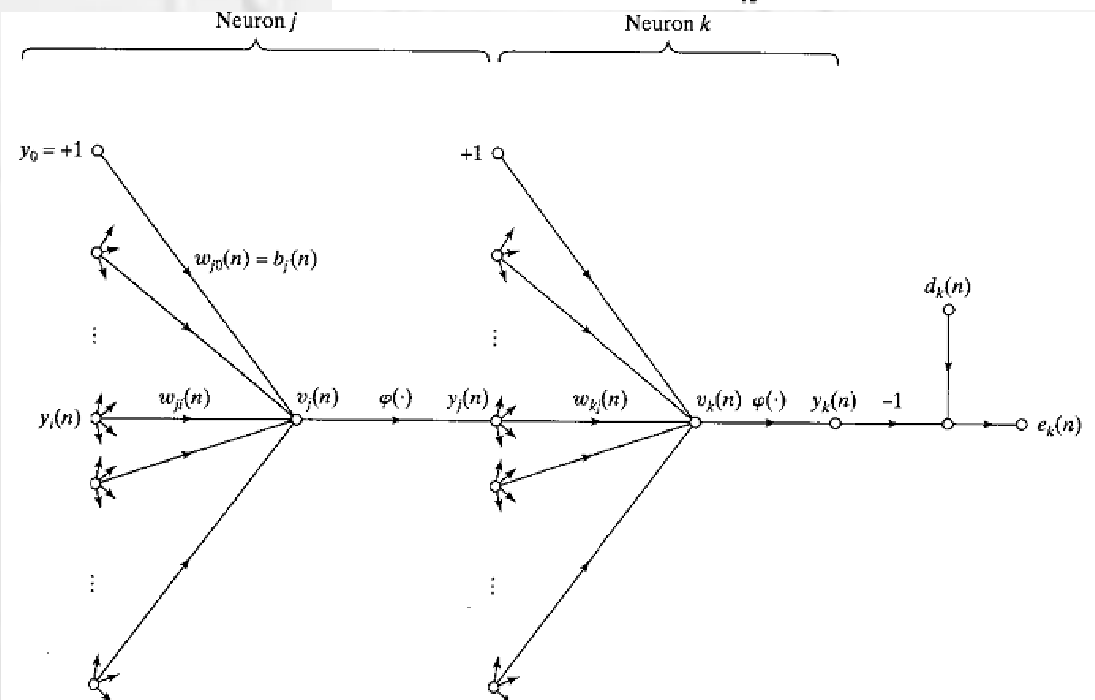
BP for Multilayer (With HN)

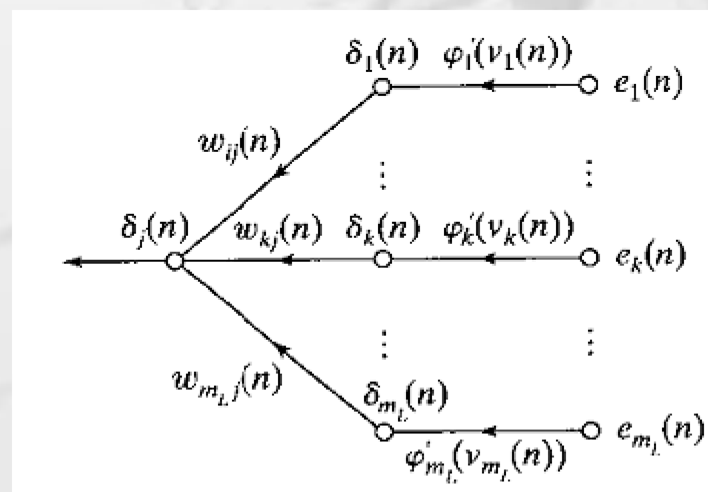
$$w_{ji}(t+1)=w_{ji}(t)+\Delta w_{ji}$$

$$\begin{pmatrix}\text{Weight}\\ \text{correction}\\ \Delta w_{ji}(n)\end{pmatrix} = \begin{pmatrix}\text{learning-}\\ \text{rate parameter}\\ \eta\end{pmatrix}\cdot\begin{pmatrix}\text{local}\\ \text{gradient}\\ \delta_j(n)\end{pmatrix}\cdot\begin{pmatrix}\text{input signal}\\ \text{of neuron } j\\ y_i(n)\end{pmatrix}$$

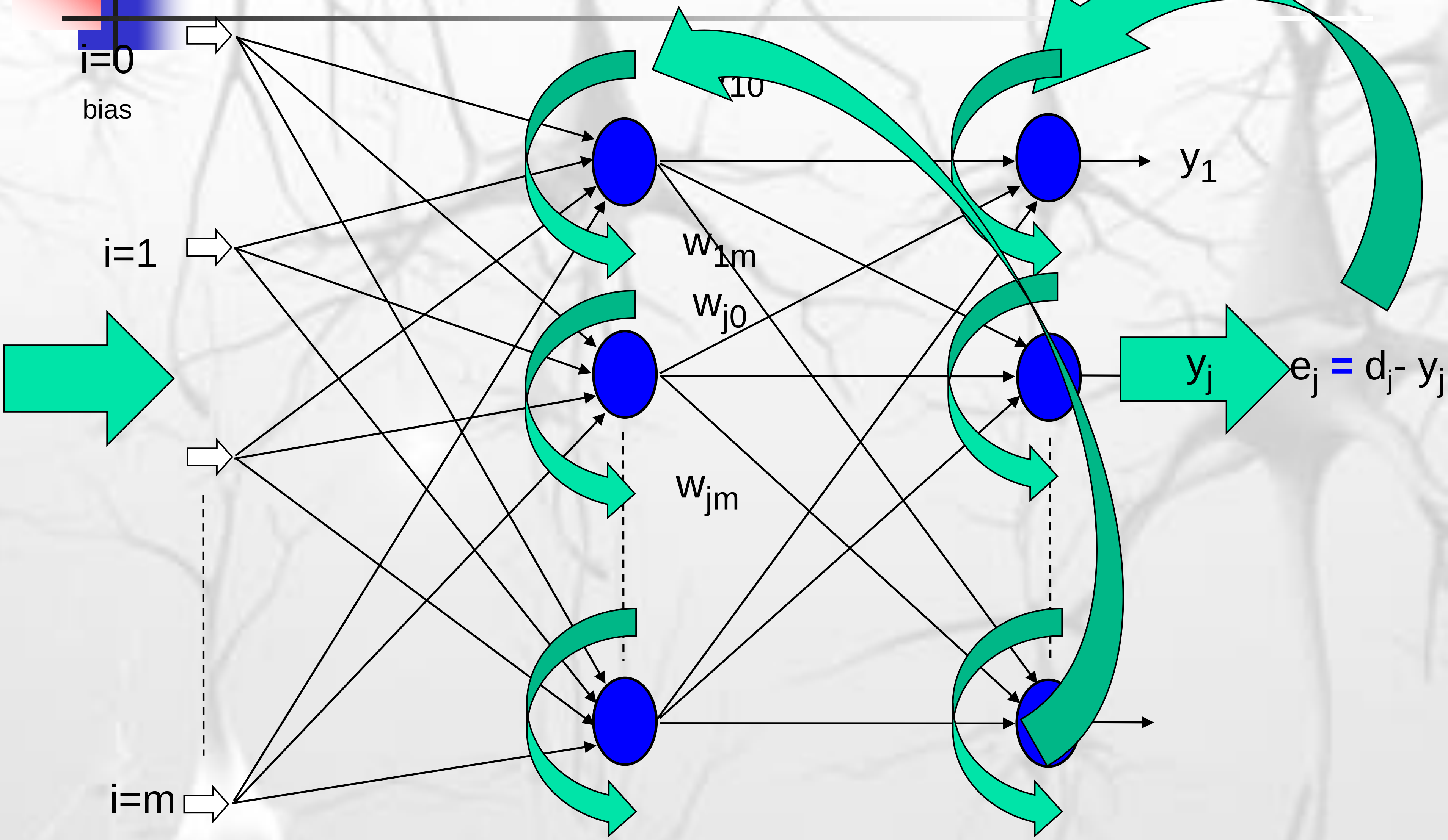
$\delta_k(n)=e_k(n)\varphi'(v_k(n))$ for k output neuron

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n), \qquad \text{neuron } j \text{ is hidden}$$

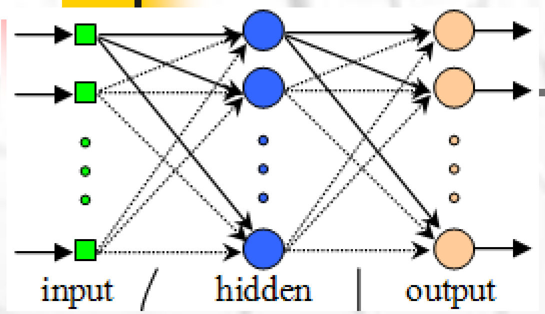




Weight Update Sequence in BP



Matter of Activation Function(AF)



$$\delta_k(n)=e_k(n)\varphi'(v_k(n))$$

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n)$$

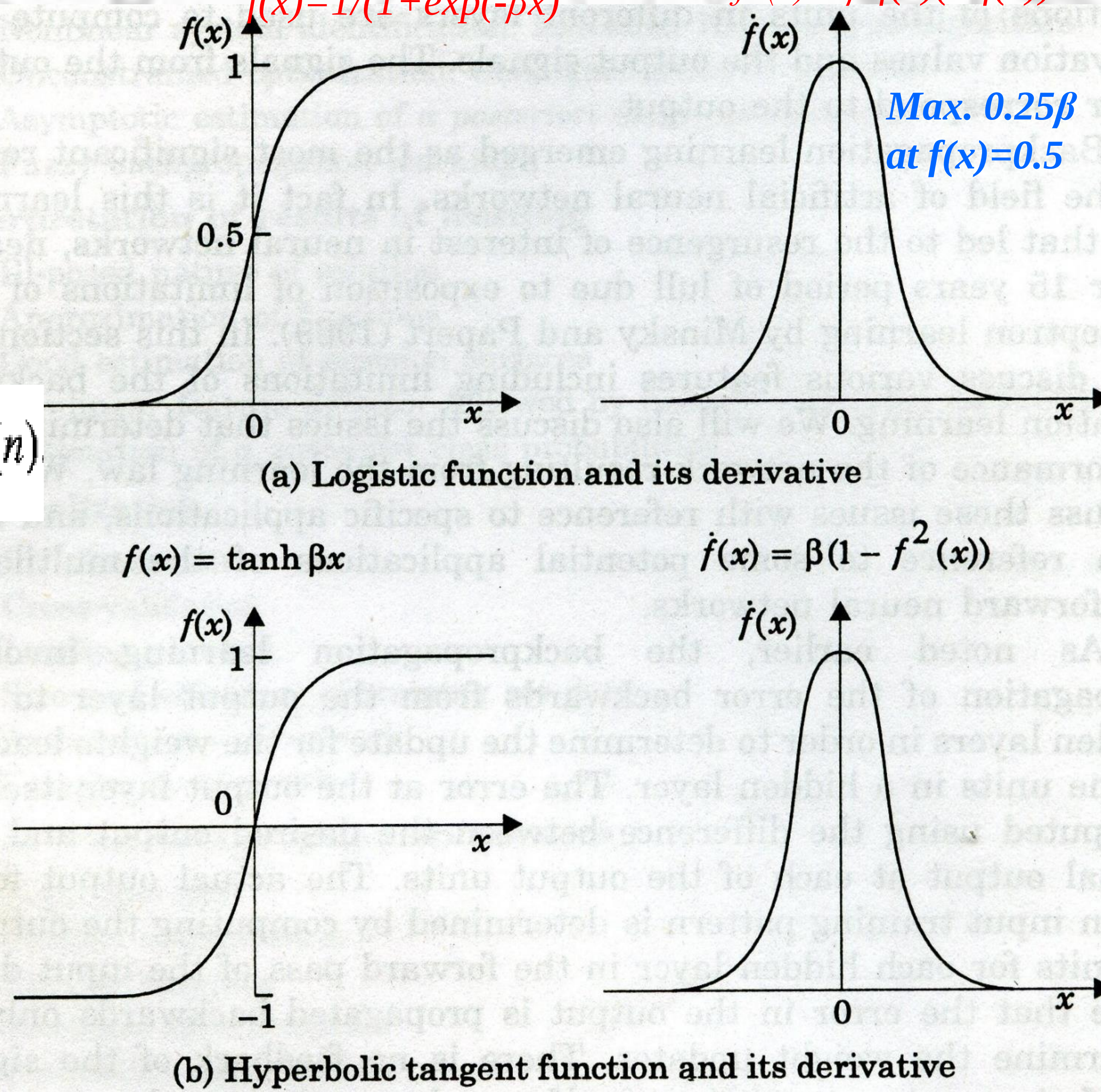
Differentiability is the only condition for AF

an example of continuously differentiable nonlinear AF is sigmoidal nonlinearity; its two forms are :

$f(x)=1/(1+exp(-\beta x))$

$f'(x)= \beta f(x)(1-f(x))$

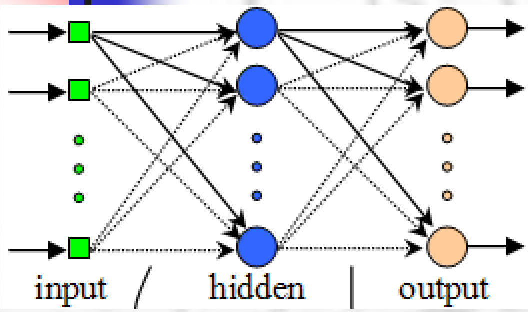
Max. 0.25β at $f(x)=0.5$



(a) Logistic function and its derivative

(b) Hyperbolic tangent function and its derivative

BP at a Glance



$$\begin{pmatrix}\text{Weight}\\ \text{correction}\end{pmatrix} = \begin{pmatrix}\text{learning-}\\ \text{rate parameter}\end{pmatrix} \cdot \begin{pmatrix}\text{local}\\ \text{gradient}\end{pmatrix} \cdot \begin{pmatrix}\text{input signal}\\ \text{of neuron } j\end{pmatrix}$$

$$\Delta w = \eta\ \delta\ x$$

The local gradient of output unit (δ_o) and hidden unit (δ_h) are defined by:

$$\delta_o = -\frac{\partial e}{\partial f_o}\frac{\partial f_o}{\partial x_o} \qquad \delta_h = \sum_o \delta_o w_o \frac{\partial f_h}{\partial x_h}$$

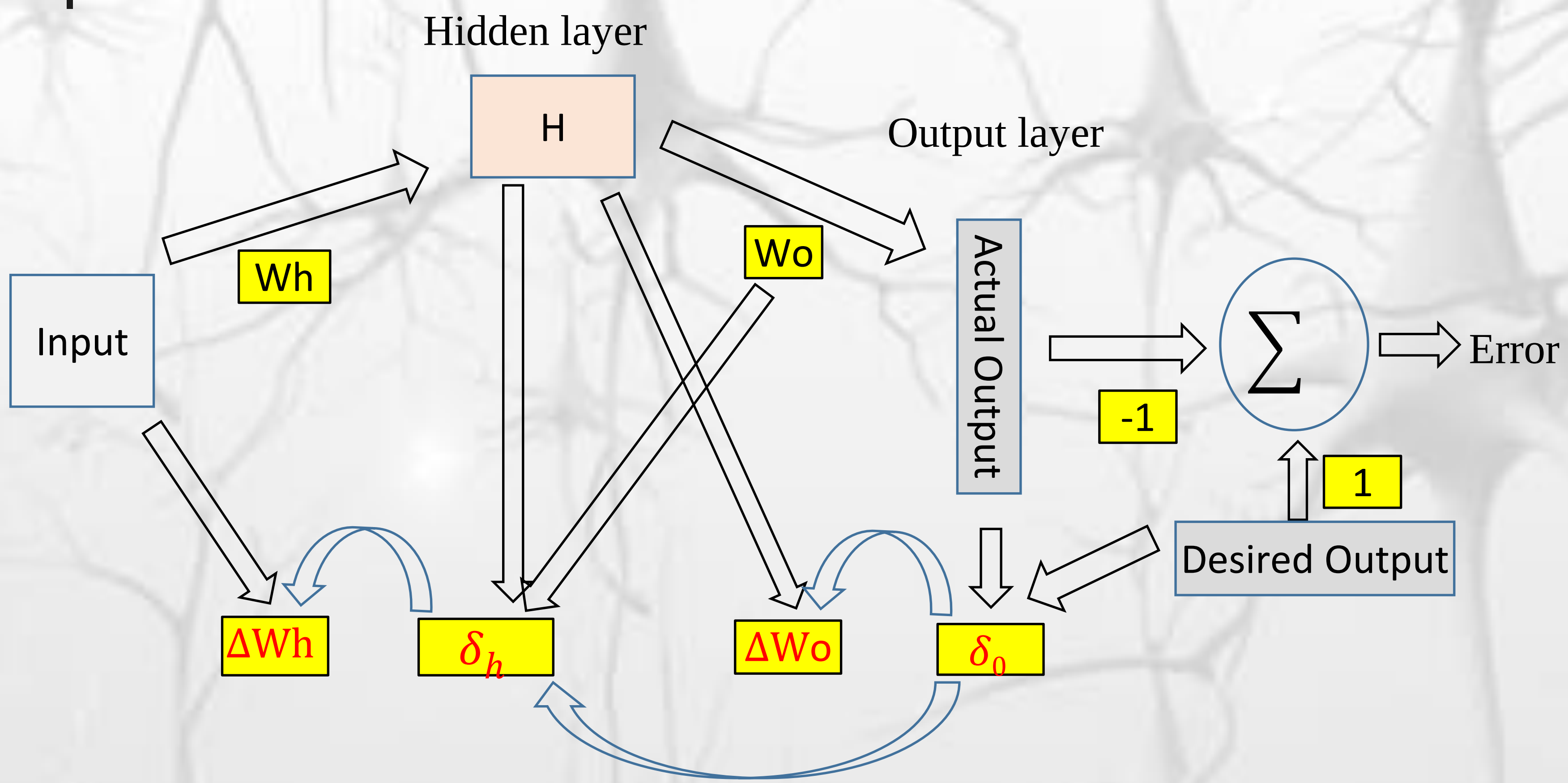
$$e(n) = \frac{1}{2}(d(n) - f_o(n))^2, \qquad \frac{\partial e_o(n)}{\partial f_o(n)} = -(d(n) - f_o(n))$$

For logistic sigmoid activation function $\quad f_o(n) = 1/(1 + \exp(-x_o(n))) \quad \frac{\partial f_o(n)}{\partial x_o(n)} = f_o(n)(1 - f_o(n))$

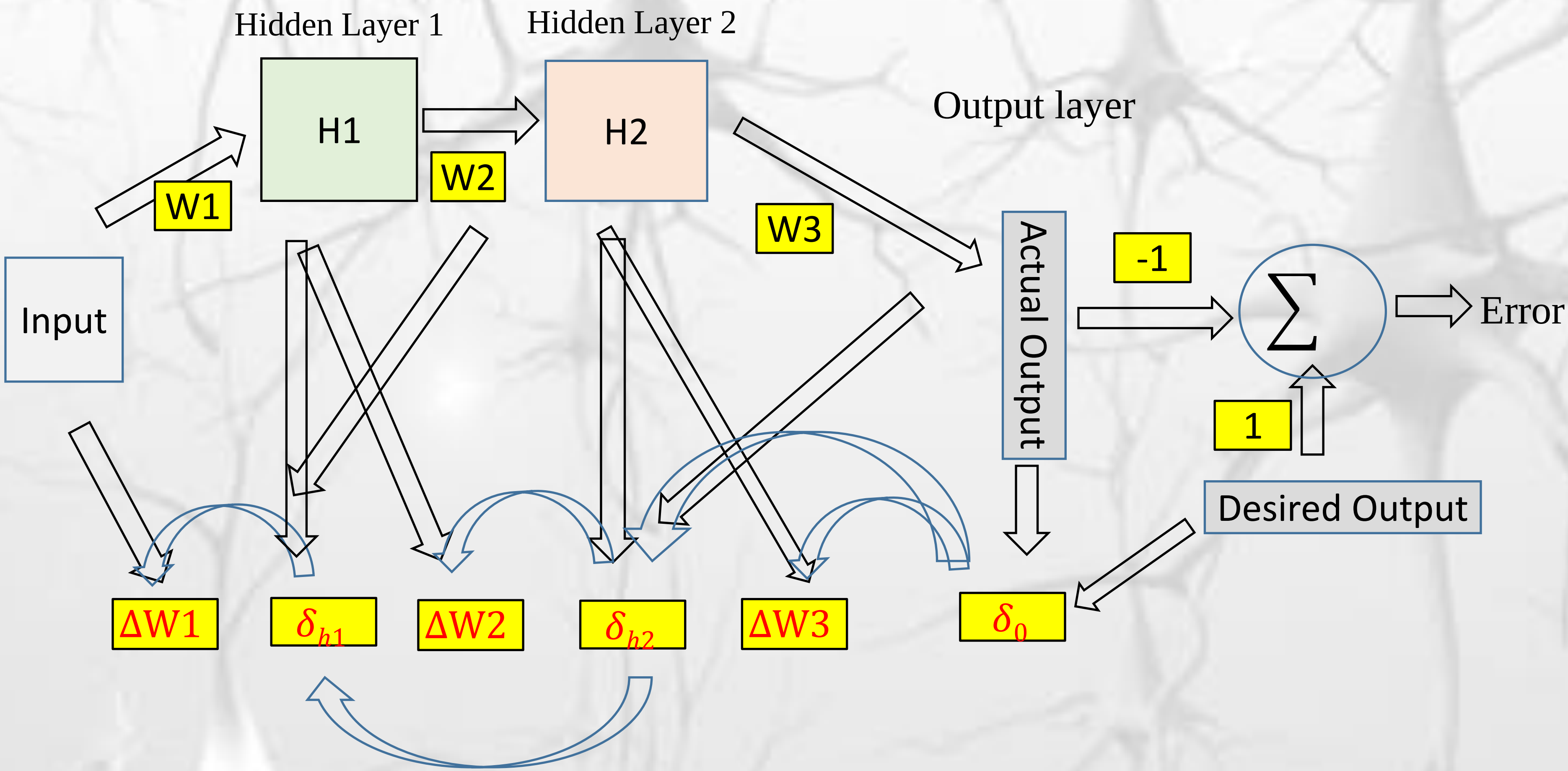
Now the local gradient of output unit (δ_o) becomes $\quad \delta_o = (d(n) - f_o(n)) f_o(n)(1 - f_o(n))$

For the same sigmoid activation function the local gradient of hidden unit (δ_h) becomes $\quad \delta_h = f_h(n)(1 - f_h(n)) \sum_o \delta_o w_o$

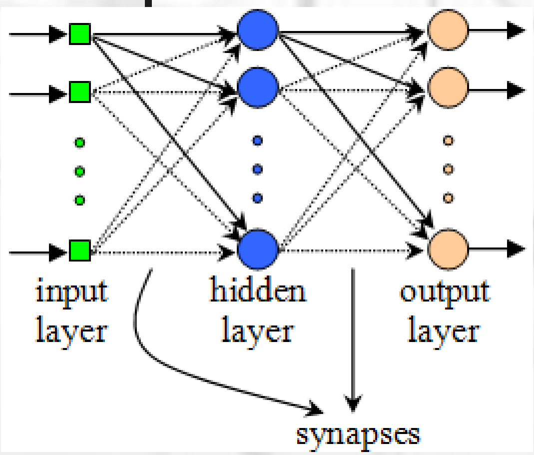
Operational flowchart of a NN with Single Hidden Layer

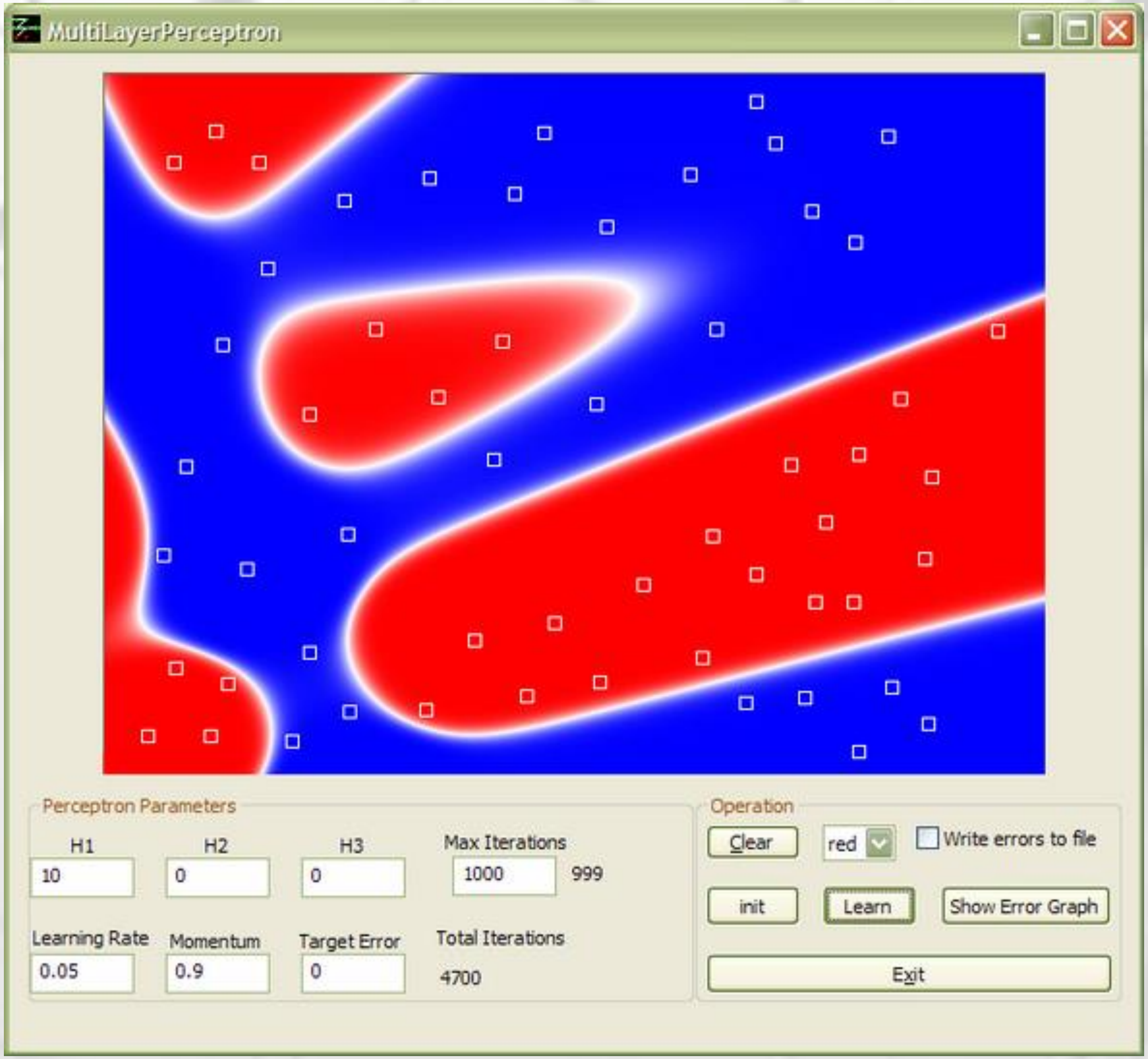


Operational flowchart of a NN with Two Hidden Layer



A MLP Simulator for Simple Application





Hossein Khosravi : The Developer of the Simulator

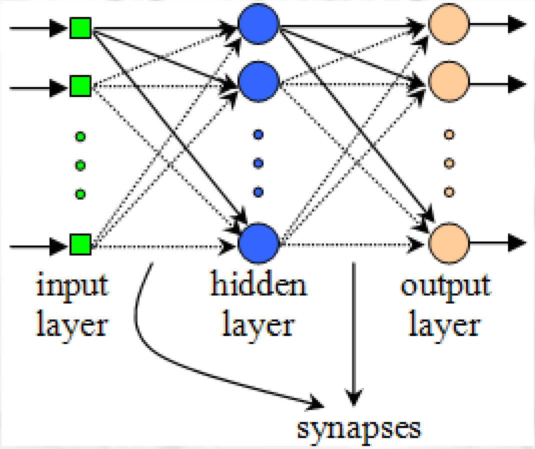
https://www.codeproject.com/Articles/9447/Neural-Network-Classifier

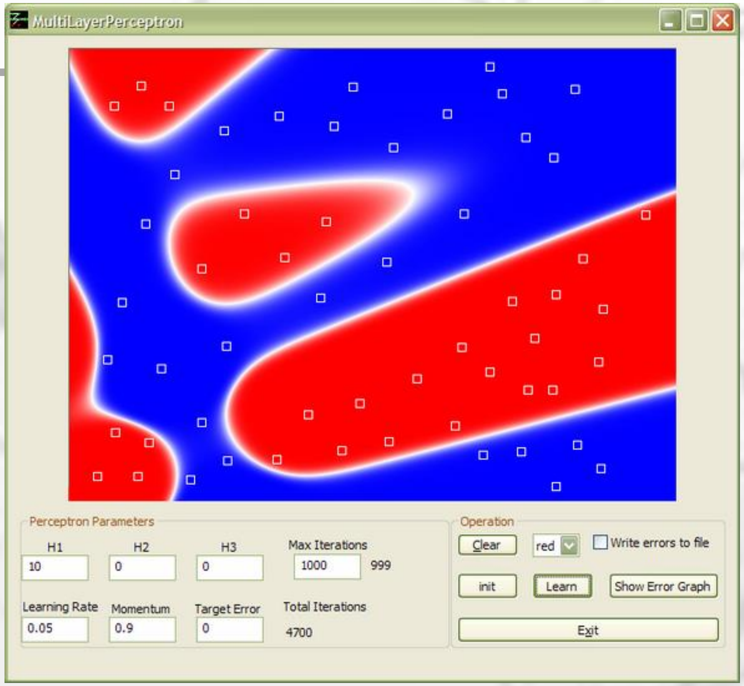
Member Profile: Hossein Khosravi

About Member

Friendly Url	http://www.codeproject.com/Members/Hossein-Khosravi
Status	**Silver**. Member No. 429414 View Member's Blog
Messages Posted	26 - Poster
Articles Submitted	2 - Contributor
Biography	I am an electronic engineer intrested in Pattern Recognition (specially OCR), Neural Networks and Image Processing. I was born in Khezri, a small town in Khorasan, the largest province of Iran, I have taken My BS. from Sharif University of thechnology and by now I'm studying in Tarbiat Modarres University as a Ph.D. candidate.
Location	Iran, Islamic Republic Of
Job Title	Web Developer
Company	
Member since	Sunday, June 08, 2003 (6 years, 4 months)
Homepage	

Blog:

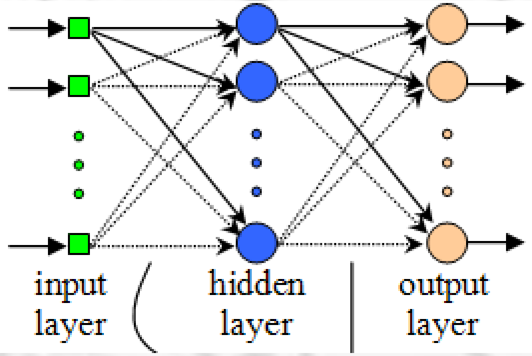




Lets run the simulator

Practical Aspects of BP

Practical Considerations



- **Training Data:** Sufficient / proper training data is require for proper input-output mapping.
- **Network Size:** Complex problem require more hidden neurons and may perform better with multiple hidden layers.
- **Weight Initialization:** NN works on numeric data. Before training all the synaptic weights are initialized with small random values, e.g., -0.5 to +0.5.
- **Learning Rate (η) and Momentum Constant (α):** A small η results slower convergence and its larger value gives oscillation. Momentum also speed up training but larger α with large η arise oscillation.
- **Stopping Training:** Training should stop at better generalization position.

Effect of Learning Rate (η) and Momentum Constant (α) values

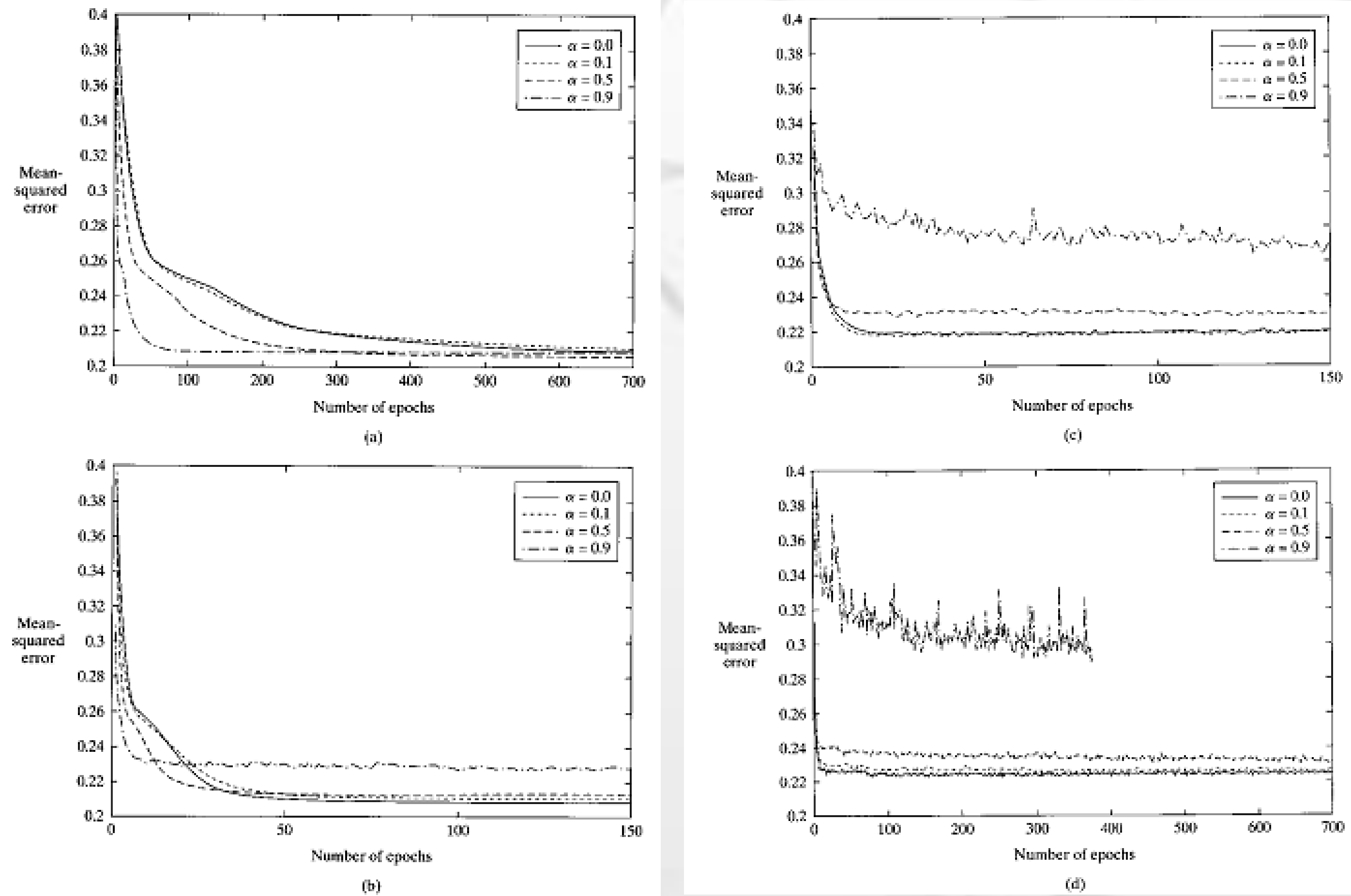


FIGURE 4.15 Ensemble-averaged learning curves for varying momentum α, and the following values of learning-rate parameters: (a) $\eta = 0.01$, (b) $\eta = 0.1$ (c) $\eta = 0.5$, and (d) $\eta = 0.9$.

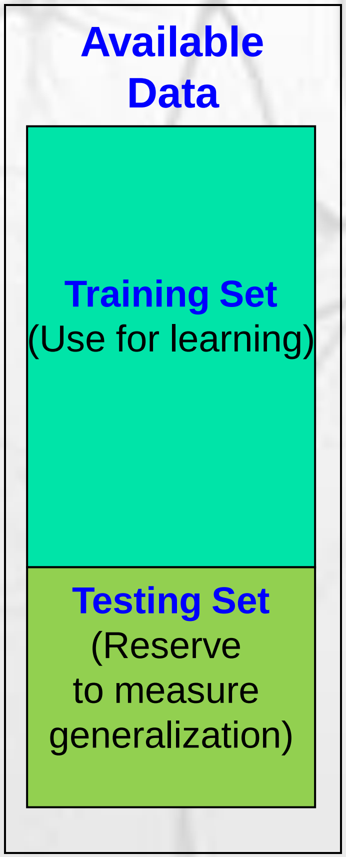
Learning and Generalization

Learning and generalization are the most important topics in NN research. Learning is the ability to approximate the training data while generalization is the ability to predict well beyond the training data.

- Generalization is more desirable because the common use of a NN is to make good prediction on new or unknown objects.
- It measures on the testing set that is reserved from available data and not use in the training.
- Testing error rate (TER), i.e., rate of wrong classification on testing set, is widely acceptable quantifying measure, which value minimum is good.

$$\text{TER} = \frac{\text{Total testing set misclassified patterns}}{\text{Total testing set patterns}}$$

Learning and Overfitting



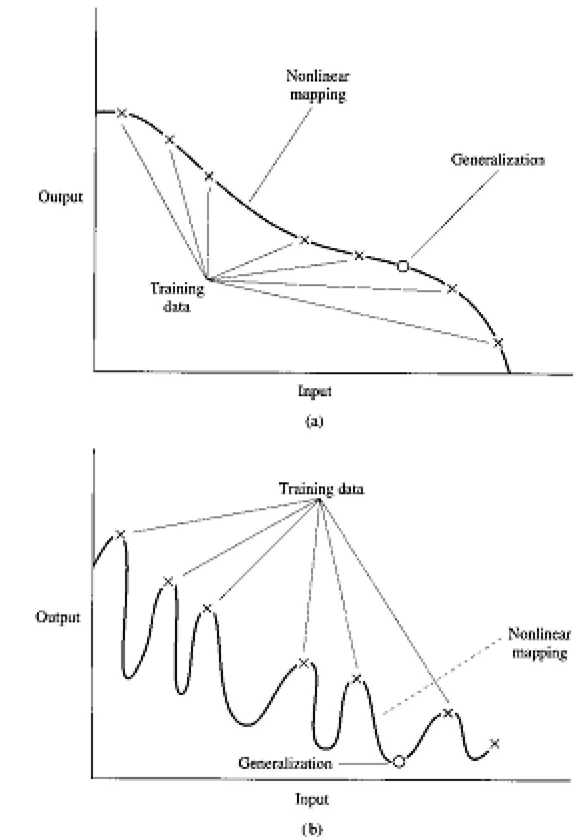
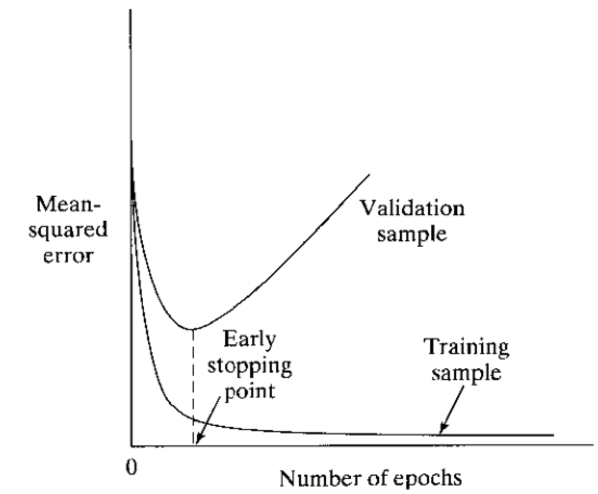
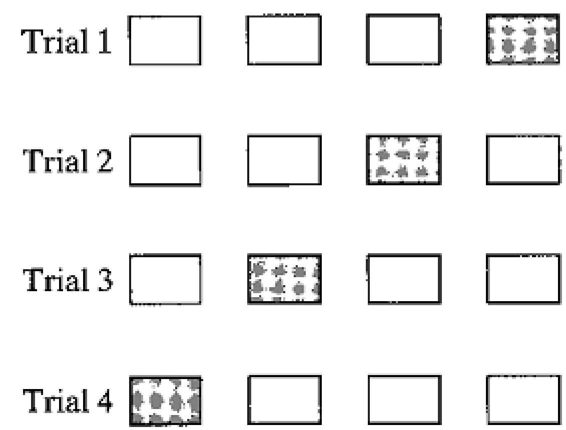


FIGURE 4.19 (a) Properly fitted data (good generalization) (b) Overfitted data (poor generalization).





Where we find benchmark data for testing NNs or machine learning?

Benchmark Problems for Evaluation

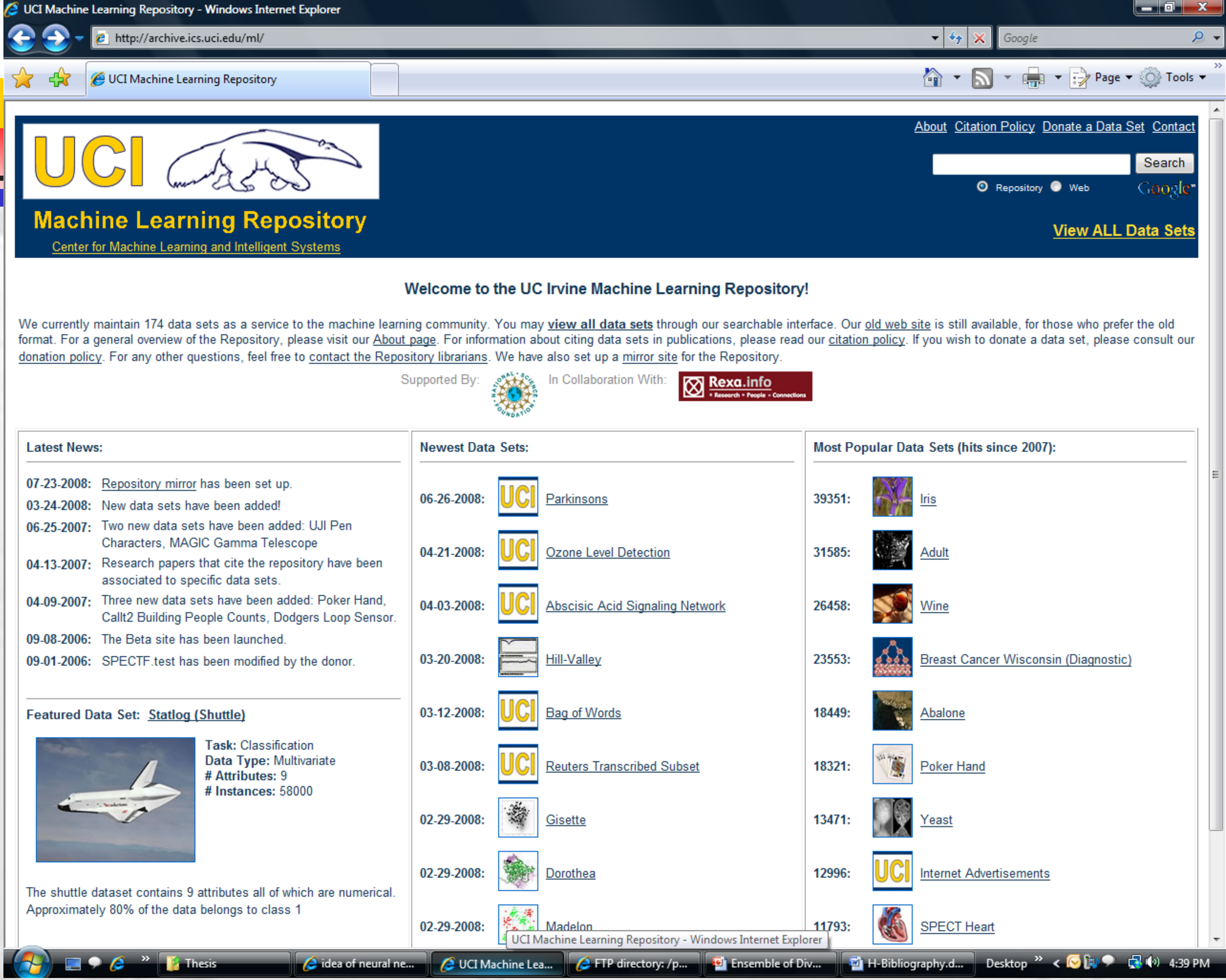
A benchmark is a point of reference by which something can be measured.

- For NN or machine learning, the most popular benchmark dataset collection is the University of California, Irvine (UCI) Machine Learning Repository (http://archive.ics.uci.edu/ml/).

- UCI contains raw data that require preprocessing to use in NN. Some preprocessed data is also available at Proben1 (ftp://ftp.ira.uka.de/pub/neuron/).

- Various persons or groups also maintain different benchmark datasets for specific purpose: Delve (www.cs.toronto.edu/~delve/data/datasets.html), Orange (www.ailab.si/orange/datasets.asp), etc.

65



UCI Machine Learning Repository - Windows Internet Explorer

http://archive.ics.uci.edu/ml/

About Citation Policy Donate a Data Set Contact

Search

Repository Web

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

View ALL Data Sets

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 174 data sets as a service to the machine learning community. You may **view all data sets** through our searchable interface. Our old web site is still available, for those who prefer the old format. For a general overview of the Repository, please visit our About page. For information about citing data sets in publications, please read our citation policy. If you wish to donate a data set, please consult our donation policy. For any other questions, feel free to contact the Repository librarians. We have also set up a mirror site for the Repository.

Supported By: In Collaboration With: Rexa.info

Latest News:

07-23-2008: Repository mirror has been set up.
03-24-2008: New data sets have been added!
06-25-2007: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope
04-13-2007: Research papers that cite the repository have been associated to specific data sets.
04-09-2007: Three new data sets have been added: Poker Hand, CalIt2 Building People Counts, Dodgers Loop Sensor.
09-08-2006: The Beta site has been launched.
09-01-2006: SPECTF.test has been modified by the donor.

Featured Data Set: Statlog (Shuttle)

Task: Classification
Data Type: Multivariate
 # Attributes: 9
 # Instances: 58000

The shuttle dataset contains 9 attributes all of which are numerical. Approximately 80% of the data belongs to class 1

Newest Data Sets:

06-26-2008: Parkinsons
04-21-2008: Ozone Level Detection
04-03-2008: Abscisic Acid Signaling Network
03-20-2008: Hill-Valley
03-12-2008: Bag of Words
03-08-2008: Reuters Transcribed Subset
02-29-2008: Gisette
02-29-2008: Dorothea
02-29-2008: Madelon

Most Popular Data Sets (hits since 2007):

39351: Iris
31585: Adult
26458: Wine
23553: Breast Cancer Wisconsin (Diagnostic)
18449: Abalone
18321: Poker Hand
13471: Yeast
12996: Internet Advertisements
11793: SPECT Heart

Benchmark Problems

Problems Related to Human Life

Problem	Task
Breast Cancer Wisconsin	Predicts whether a tumor is benign (not dangerous to health) or malignant (dangerous) based on a sample tissue taken from a patient’s breast.
BUPA Liver Disorder	Identify lever disorders based on blood tests along with other related information such as alcohol consumption.
Diabetes	Investigate whether the patient shows or not the signs of diabetes.
Heart Disease Cleveland	Predicting whether at least one of four major heart vessels is reduced in diameter by more than 50%.
Hepatitis	Anticipate status (i.e., live or die) of hepatitis patient.
Lymphography	Predict the situation of lymph nodes and lymphatic vessels.
Lungcancer	Identify types of pathological lung cancers.
Postoperative	Determine place to send patients for postoperative recovery.

Benchmark Problems

Problems Related to Finance

Problem	Task
Australian Credit Card	Classify people as good or bad credit risks depend on applicants' particulars.
Car	Evaluate cars based on price and facilities.
Labor Negotiations	Identify a worker as good or bad i.e., contract with him beneficial or not.
German Credit Card	Like Australian Card, this problem also concerns to predict the approval or non-approval of a credit card to a customer.

Problems Related to Plants

Problem	Task
Iris Plants	Classify iris plant types.
Mushroom	Identify whether a mushroom is edible or not based on a description of the mushroom's shape, color, odor, and habitat.
Soybean	Recognize 19 different diseases of soybeans.

Benchmark Problems and NN Architecture

After Preprocessing

Depends on Problem

Abbr.	Problem	Total Examp	Input Features		NN Architecture		
			Cont.	Discr.	Inputs	Class	Hidd. Node
ACC	Australian Credit Card	690	6	9	51	2	10
BLN	Balance	625	-	4	20	3	10
BCW	Breast Cancer Wisconsin	699	9	-	9	2	5
CAR	Car	1728	-	6	21	4	10
DBT	Diabetes	768	8	-	8	2	5
GCC	German Credit Card	1000	7	13	63	2	10
HDC	Heart Disease Cleveland	303	6	7	35	2	5
HPT	Hepatitis (HPT)	155	6	13	19	2	5
HTR	Hypothyroid	7200	6	15	21	3	5
HSV	House Vote	435	-	16	16	2	5
INS	Ionosphere	351	34	-	34	2	10
KRP	King+Rook vs King+Pawn	3196	-	36	74	2	10
LMP	Lymphography	148	-	18	18	4	10
PST	Postoperative	90	1	7	19	3	5
SBN	Soybean	683	-	35	82	19	25
SNR	Sonar	208	60	-	60	2	10
SPL	Splice Junction	3175	-	60	60	3	10
WIN	Wine	178	13	-	13	3	5
WVF	Waveform	5000	21	-	21	3	10
ZOO	Zoo	101	15	1	16	7	10

Input Features of Diabetes

1. Number of times pregnant
2. Plasma glucose concentration
3. Diastolic blood pressure
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index
7. Diabetes pedigree function
8. Age

❖Problems show variations in number of examples, input features and classes.

An Application of Feed Forward Neural Network

Optical Character Recognition(OCR)

Optical Character Recognition(OCR)

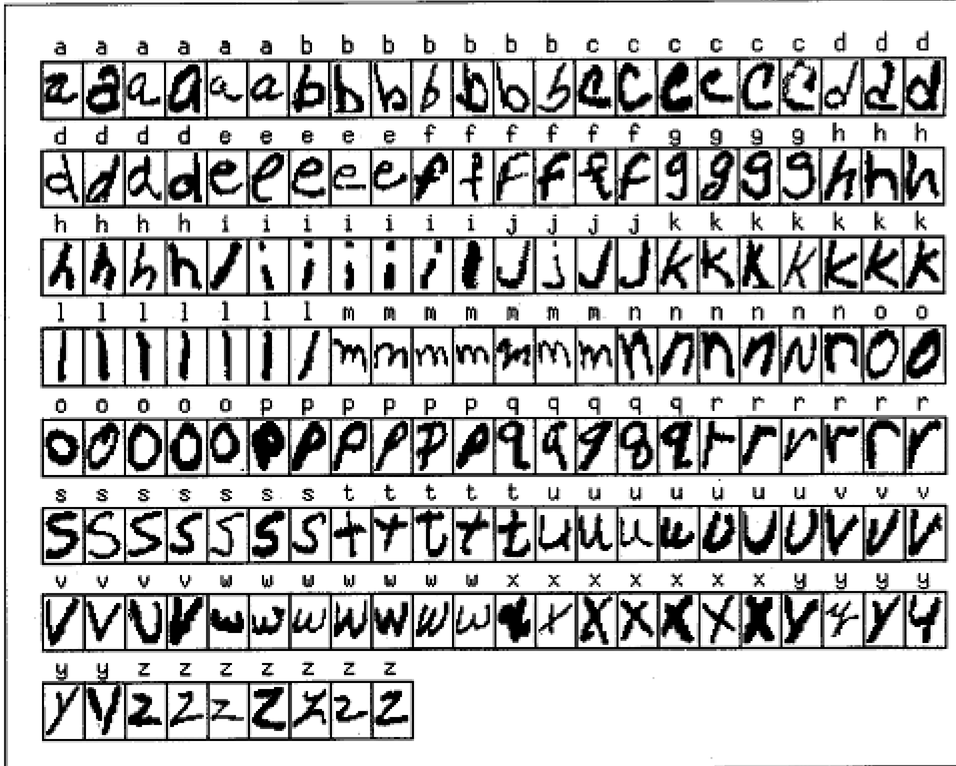


Figure 10. A sample set of characters in the NIST database.

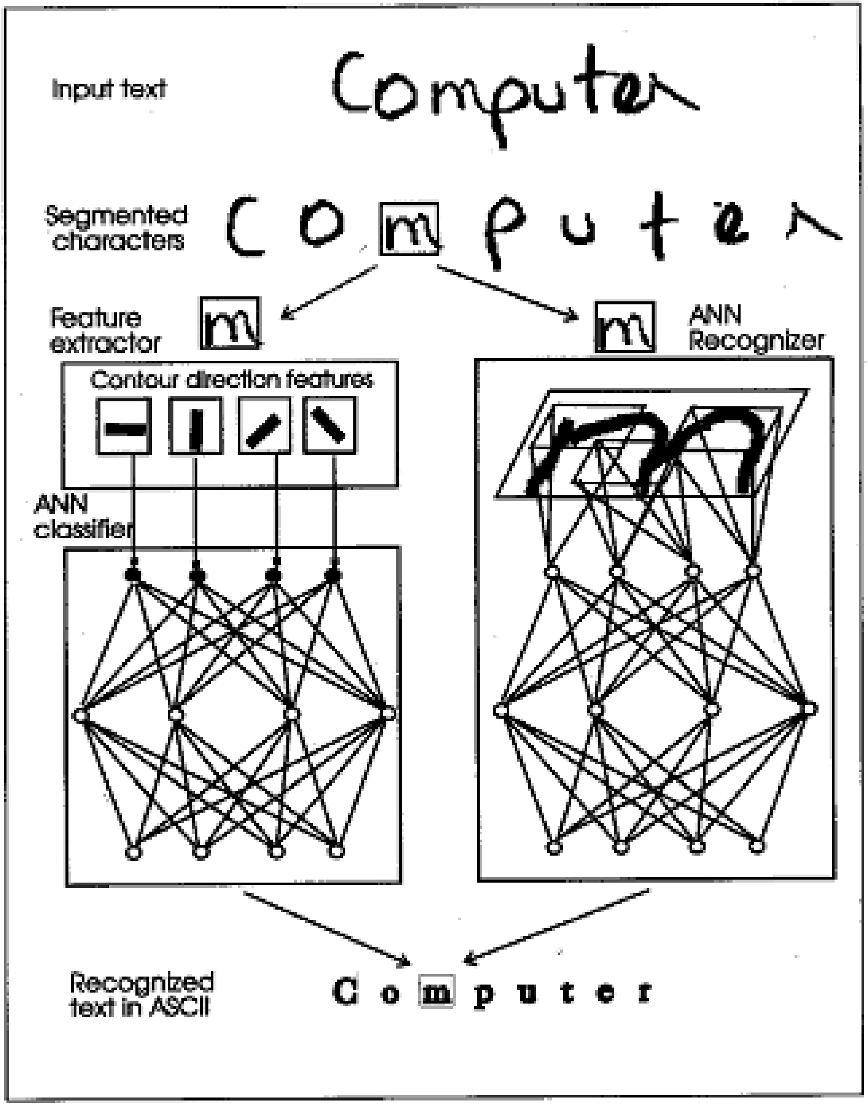


Figure 11. Two schemes for using ANNs in an OCR system.